ВСТУП ДО КРИПТОЛОГІЇ

О. В. Вербіцький

ВИДАВНИЦТВО НАУКОВО-ТЕХНІЧНОЇ ЛІТЕРАТУРИ
Львів • 1998

Видання пропонується всім, хто зацікавлений у знайомстві з предметом. Кожен розділ, присвячений певному класові криптосистем, починається обговоренням основних концепцій і містить опис конкретних алгоритмів та їх детальний математичний аналіз. Виклад математичного апарату максимально замкнений. Посібник містить багато вправ і може бути основою одно- або, залежно від рівня математичної підготовки аудиторії, двосеместрового курсу з основ захисту інформації.

ВНТЛ — Видавництво науково-технічної літератури

Адреса офісу: 290007, м.Львів, вул. Огієнка 18-а, оф. 1.3.
Адреса для листування: 290007, м. Львів, а/с 1173
тел./факс: (0322) 728073
e-mail: vntl@litech.lviv.ua
http://www.litech.lviv.ua/~vntl

Художній редактор — Леся Дуб

В оформленні обкладинки використано текст з роману
Юрія Андруховича «Перверзії»

ISBN 966-7148-03-3
ISBN 966-7148-23-8 (Серія «Університетська математика: спеціальні курси»)

Зміст

Передмова 8

Про побудову книги 10

Розділ I. **Елементарна криптографія** 12

§ 1. Абетка 12

1.1. Початкові поняття та приклади (12). 1.2. Термінологічні застереження (15). 1.3. Ретроспективні зауваження (16). Вправи (18). Література (18).

§ 2. Класичні методи: від античності до нового часу 19

2.1. Шифри простої заміни (19). 2.2. Частотний аналіз (20). 2.3. Поліграмні шифри (22). 2.4. Поліалфавітні шифри (23). 2.5. Шифрування блоками (26). 2.6. Шифри перестановки (26). Вправи (30). Література (34).

§ 3. Дві пропозиції XX століття 34

3.1. Подання тексту у цифровій формі (35). 3.2. Шифр одноразового блокноту (35). 3.3. Кількаразове шифрування (37). 3.4. DES (40). 3.5. Рецепти використання блокових шифрів (43). Вправи (44). Література (45).

Розділ II. **Елементарна криптографія II: математичний підхід** 46

§ 1. Формалізм 46

1.1. Абетка II (46). 1.2. Дешифрування ітераціями (48). Вправи (49).

§ 2. Арифметика 49

2.1. Алгоритм Евкліда (49). 2.2. Розклад на прості співмножники (51). 2.3. Конгруенції (52). 2.4. Кільце лишків (53). 2.5. Кільце матриць (56). Вправи (58). Література (59).

§ 3. Афінні шифри 60

3.1. Шифри простої заміни II (60). 3.2. Афінні шифри вищих порядків (62). 3.3. Криптоаналіз (65). Вправи (66). Література (71).

Розділ III. **Алгоритми та їх складність** 72

§ 1. Поняття та терміни 72

1.1. Задачі (72). 1.2. Алгоритми (74). Вправи (77). Література (77).

§ 2. Прямолінійні програми 78

2.1. Означення моделі (78). 2.2. Піднесення до степеня (79). 2.3. Функціональні схеми (80). Вправи (82). Література (83).

§ 3. Рандомізація 84

3.1. Ймовірнісні алгоритми (84). 3.2. Випадковий вибір (85). Вправи (87).

§ 4. Порівняння складності задач 89

4.1. Оракульна модель (89). 4.2. Звідності задач (90). Вправи (92).

Розділ IV. **Складність арифметичних задач** 93

§ 1. Арифметика II 93

1.1. Первісні корені (94). 1.2. Квадратичні лишки (95). 1.3. Розподіл простих чисел (98). Вправи (98). Література (100).

§ 2. Тестування простоти 100

2.1. Ймовірнісний тест Соловея-Штрассена (102). 2.2. Лас Вегас алгоритм (103). 2.3. Псевдопрості числа (104). 2.4. Ймовірнісний тест Міллера-Рабіна (105). 2.5. Прості числа до 25 000 000 000 (105). 2.6. Розширена гіпотеза Рімана (106). 2.7. Виробництво простих чисел (107). Вправи (108). Література (109).

§ 3. Факторизація 109

Вправи (110). Література (111).

§ 4. Розпізнавання квадратичності і добування квадратних коренів 111

4.1. Квадратичність у випадку простого модуля (111). 4.2. Добування квадратного кореня за простим модулем (112). 4.3. Випадок модуля $n = pq$ (113). Вправи (116). Література (117).

§ 5. Обчислення функції Ойлера 118

5.1. Випадок аргументу $n = pq$ (118). 5.2. Деякі узагальнення (118). Вправи (121). Література (121).

§ 6. Первісні корені за простим модулем 121

6.1. Розпізнавання (121). 6.2. Породження (122). Вправи (123). Література (123).

§ 7. Дискретний логарифм 123

7.1. Складність дискретного логарифмування (123). 7.2. Алгоритм Сільвера-Поліга-Гелмана (124). Вправи (125). Література (125).

§ 8. Підсумок 126

Розділ V. **Криптосистеми з відкритим ключем** 127

§ 1. Концепція 127

Література (130).

§ 2. RSA 130

2.1. Опис системи (130). 2.2. Коректність (131). 2.3. Ефективність (132). 2.4. Надійність (132). Вправи (136). Література (137).

§ 3. Система Рабіна 137

Вправи (139).

§ 4. Ймовірнісне криптування 139

4.1. Ідея (139). 4.2. Реалізація на основі квадратичності (140). 4.3. Реалізація на основі RSA функції (142). Вправи (143).

§ 5. Система ЕльГамала 143

Вправи (145).

Розділ VI. **Криптографічні інструменти** 146

§ 1. Важкооборотні функції 146

1.1. Означення і приклади (146). 1.2. Перші застосування (147). 1.3. Поняття ядра функції (149). 1.4. Предикат із секретом і ймовірнісне криптування (150). Вправи (151). Література (152).

§ 2. Генератори псевдовипадкових бітів 152

2.1. Означення генератора псевдовипадкових послідовностей (152). 2.2. Непередбачуваність псевдовипадкових генераторів (154). 2.3. Псевдовипадкові генератори із важкооборотних перестановок (155). 2.4. BBS генератор (156). 2.5. Потокові шифри (158). Вправи (159). Література (161).

Розділ VII. **Протоколи** 162

§ 1. Обмін ключем 162

Вправи (164). Література (164).

§ 2. Цифровий підпис 164

2.1. Підпис у системі RSA (164). 2.2. Загальна схема (166). 2.3. Система цифрового підпису ЕльГамала (167). 2.4. Коди достовірності (168). 2.5. Система Шнорра (170). 2.6. DSA (171). Вправи (172). Література (173).

§ 3. Підкидання монети по телефону 174

3.1. Попереднє обговорення (174). 3.2. Протокол на основі ймовірнісного криптування (175). 3.3. Протокол на основі дволистої функції із секретом (176). Вправи (176). Література (177).

§ 4. Гра в карти заочно 177

Вправи (178). Література (179).

§ 5. Розподіл таємниці 179

Вправи (180).

§ 6. Доведення без розголошення 180

6.1. Доведення квадратичності (180). 6.2. Доведення неквадратичності (183). Вправи (184). Література (184).

§ 7. Ідентифікація 185

7.1. Ідентифікація за допомогою симетричної криптосистеми (185). 7.2. Ідентифікація на підставі цифрового підпису (185). 7.3. Ідентифікація як доведення без розголошення (185). Вправи (186). Література (186).

Розділ VIII. **Питання $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ і криптографія** 187

§ 1. Клас $\mathcal{NP}$ 187

1.1. Недетерміновані обчислення (187). 1.2. Приклади задач з класу $\mathcal{NP}$ (188). 1.3. Питання $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ (188). 1.4. $\mathcal{NP}$-повнота (189). 1.5. Приклади $\mathcal{NP}$-повних задач (190). Вправи (190). Література (191).

§ 2. Доведення без розголошення для мов у класі $\mathcal{NP}$ 191

Вправи (195). Література (195).

Додаток А. **Таблички частот** 196

Додаток Б. **Математичний апарат** 198

Б.1. Множини (198). Б.2. Відображення (198). Б.3. Групи (199). Б.4. Група перестановок (201). Б.5. Кільця (201). Б.6. Кільце матриць (203). Б.7. Поля (204). Б.8. Кільце многочленів (205). Б.9. Векторні простори (206). Література (207).

Додаток В. **Колекція нерівностей** 208

Додаток Г. **Ключ до вправ** 212

Бібліографія 229

Предметний покажчик 239

Романові Самборському

Передмова

В. пояснив свій таємний проект. Він нікому про це не говорив. Останні тридцять три роки (це — містичне число) він намагався проникнути в шифр. Він вивчив усе. Він прочитав усе. Не лише родинні архіви, але й Спінозу, і Мандевілля, і Йоахіма з Фйори.

Аскольд Мельничук, "Дерево світла".

Криптологія охоплює криптографію — науку про збереження таємниці тексту, та криптоаналіз — науку про проникнення у таємницю захищеного тексту. Із появою у 1976 році ідеології відкритого ключа криптографічна практика почала використовувати фундаментальні результати теорії чисел і одночасно стала джерелом нових глибоких математичних задач. Як наслідок, на сьогоднішній день криптологія перетворилась у респектабельну математичну дисципліну з класичною структурою: означення — теорема — доведення. Поняття надійності криптосистеми отримало чіткі означення, і надійність чи ненадійність тої чи іншої криптосистеми стала доводиться як теорема.

Цей підручник пропонується всім, хто зацікавлений в ознайомленні з предметом. Він має вступний характер, з огляду на який значна увага відводиться конкретним криптосистемам — від найдавніших до наймодерніших. На такому суто практичному матеріалі ілюструються загальні ідеї та концепції сучасної криптології.

Одним із завдань, які я ставив перед собою, було вписати криптологію в систему університетських математичних дисциплін. Підручник виник на основі семестрових курсів, які я читав на механіко-математичному факультеті Львівського університету впродовж 1996–97 років. Задля справедливості слід зазначити, що в той час, як у нас в цьому напрямку робляться лише перші кроки, курс з основ захисту інформації вже давно посів своє місце у прикладній математичній освіті на Заході, а віднедавна — і на прогресуючому Сході.

Одночасно я намагався зробити цей посібник доступним для якомога ширшого читацького кола, передусім за рахунок чіткої структурованості викладу. Практично орієнтований читач легко відшукає

описи стандартів шифрування DES або цифрового підпису DSS, а читач, якого цікавлять математичні аспекти, заглибиться у криптоаналіз системи RSA або зосередиться на конструкції генератора псевдовипадкових бітів на основі довільної важкооборотної функції із секретом. Сподіваюся, що кожен лектор отримає досить матеріалу для компонування власного курсу із врахуванням математичної підготовки своєї аудиторії. Мій досвід свідчить, що рівень цієї підготовки може бути дуже різним. Досить згадати, що криптографія є традиційною темою у популярній математиці. Сам я доволі вагому частину поданого в посібнику матеріалу, включно із криптографією відкритого ключа, з обопільним задоволенням розповідав школярам на заняттях Львівської Малої Академії Наук.

Жоден із результатів у цьому підручнику не є моїм власним. Підбір матеріалу в його суто криптографічній частині є традиційним, а його організація і включення у відповідний математичний контекст великою мірою відображають мій особистий викладацький досвід та наукові вподобання.

Приємним обов'язком для мене є згадати тих, у багатолітньому спілкуванні з ким були сформовані мої наукові інтереси і смаки — О. О. Разборова, М. К. Верещагіна та О. Х. Шеня. Цінним досвідом було співробітництво з лабораторією з математичних проблем криптографії Московського університету, а саме, з М. П. Варновським, В. М. Сидельниковим та В. В. Ященком. У роботі над книгою відчутну допомогу надали Р. В. Ардан, І. Я. Берегуляк, Я. Б. Воробець, О. О. Гринів, Р. О. Гринів, В. В. Ігнатюк, З. В. Партико та Я. М. Холявка. Особлива вдячність В. І. Дмитеркові, М. М. Зарічному та Я. Г. Притулі, без ентузіазму і підтримки яких цей посібник ще довго залишався б нереалізованим проектом.

Тернопіль, серпень 1997 *Олег Вербіцький*

Про побудову книги

Розділ I є зовсім елементарним. Він охоплює класичну криптографію — від давньогрецького шифру *Скитала* до стандарту DES, ухваленого в Сполучених Штатах у 1976 році. В цьому розділі вводяться поняття і терміни, які далі використовуються в усьому курсі. В розділі II класичні шифри вивчаються на формальному математичному рівні як часткові випадки афінного шифру.

Розділ III присвячено базовим поняттям теорії складності обчислень. У мінімальний курс з цього розділу обов'язково мусить ввійти бінарний алгоритм піднесення до степеня (пункт 2.2) та метод пониження однобічної помилки ймовірнісного алгоритму (пункт 3.1).

Розділ IV містить систематизований виклад алгоритмів арифметики кільця $\mathbb{Z}_n$ та класифікацію арифметичних задач за їх складністю. На цей розділ робиться чимало посилань у подальшому викладі криптографії відкритого ключа в розділах V (шифри) та VII (протоколи).

У розділі VI йдеться про важкооборотні функції і генератори псевдовипадкових бітів, а також про потокові шифри. При висвітленні цих питань ми свідомо не вдаємося в технічні тонкощі, а більше акцентуємо увагу на ідейній стороні справи. Останній розділ VIII має оглядовий характер. Він висвітлює взаємозв'язки між криптографією та теорією складності, а саме, концепцією $\mathcal{NP}$-повноти.

Кожен розділ складається з кількох параграфів. Параграфи закінчуються вправами та оглядом літератури на відповідну тему. Вправи можуть суттєво відрізнятися між собою за складністю — від простих обчислювальних прикладів до важливих теоретичних результатів. Для більшості вправ у додатку Г наведено відповіді та розв'язання.

У додатку Б зібрано разом факти з алгебри і теорії чисел, які використовуються в курсі.

У предметному покажчику в кінці книги кожен криптологічний термін подається разом зі своїм англійським відповідником. Слід зауважити, що пропоновані в цьому посібнику українські терміни не завжди є прямими перекладами англійських.

Більшість параграфів розбито на пункти, для яких прийнято подвійну нумерацію. Наприклад, § 4 розділу III складається з пунктів 4.1 та 4.2. При посиланні з іншого розділу ці пункти згадуються під номерами III.4.1 та III.4.2. Подібним чином нумеруються теореми, означення, зауваження та приклади.

Для того, щоб читач мав можливість в усіх деталях слідкувати за прикладами шифрування та дешифрування, ми наводимо тут українську, російську та латинську абетки. Кожна літера супроводжується своїм порядковим номером в абетці, причому нумерація починається з нуля.

Українська абетка

А	Б	В	Г	Ґ	Д	Е	Є	Ж	З	И
0	1	2	3	4	5	6	7	8	9	10
І	Ї	Й	К	Л	М	Н	О	П	Р	С
11	12	13	14	15	16	17	18	19	20	21
Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я
22	23	24	25	26	27	28	29	30	31	32

Російська абетка

А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й
0	1	2	3	4	5	6	7	8	9	10
К	Л	М	Н	О	П	Р	С	Т	У	Ф
11	12	13	14	15	16	17	18	19	20	21
Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
22	23	24	25	26	27	28	29	30	31	32

Латинська абетка

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Розділ I.

Елементарна криптографія

§ 1. Абетка

1.1. Початкові поняття та приклади. Класична задача криптографії постає тоді, коли двоє збираються обмінятися конфіденційною інформацією за присутності третьої недружньої персони, яка також намагається заволодіти цією інформацією. Підтримуючи усталену у криптографічній літературі традицію, ми назвемо двох перших осіб Алісою та Бобом. Третій персонаж називатиметься *суперником*. Аліса та Боб є законними користувачами каналу зв'язку або легальними абонентами комунікаційної мережі. Суперник є несанкціонованим користувачем, який має нешляхетну мету перехопити чи підслухати Бобове повідомлення Алісі. Аби зберегти таємницю, Боб *шифрує* своє повідомлення, тобто перетворює його до незрозумілої для суперника форми, застосувавши *алгоритм шифрування*. В результаті з повідомлення, яке ще називають *відкритим текстом*, виходить *криптотекст*, який Боб і посилає Алісі. Отримавши криптотекст, Аліса *дешифрує* його за допомогою *алгоритму дешифрування* і отримує вихідне повідомлення. В наших криптологічних студіях ми виходимо із припущення, що суперникові завжди щастить перехопити криптотекст. Проте якщо алгоритм шифрування є *надійним*, то суперник не здатен дошукатися змісту повідомлення.

Алгоритми шифрування та дешифрування разом складають *криптосистему* або, простіше, *шифр*.

Розглянемо приклад. Поставивши себе на місце суперника, уявимо що нам вдалося підслухати зашифроване повідомлення:

осіла унофелет од шидохдіп ен умоч

Спробуймо відновити повідомлення, відгадавши алгоритм шифрування. Поза сумнівом, успіх буде миттєвим[1].

Шифр, з яким ми щойно ознайомились, має дві типові риси, бажану і небажану. Перша полягає в тому, що цей шифр є *ефективним* — шифрування і дешифрування займають зовсім мало часу. Негативною рисою цього шифру є його *ненадійність*. Найчастіше вимоги, які висуваються до криптосистеми, не вдається задовольнити одночасно — ідеального шифру не існує. Вибір шифру з тими чи іншими властивостями диктується конкретною ситуацією. Іноді інформація, скажімо біржова, перестає бути таємною через двадцять хвилин і мусить бути зашифрована і передана за лічені секунди. А іноді інформація повинна зберігатись десятиліттями, зате нема потреби квапитись при шифруванні.

Розглянемо ще дві прості криптосистеми.

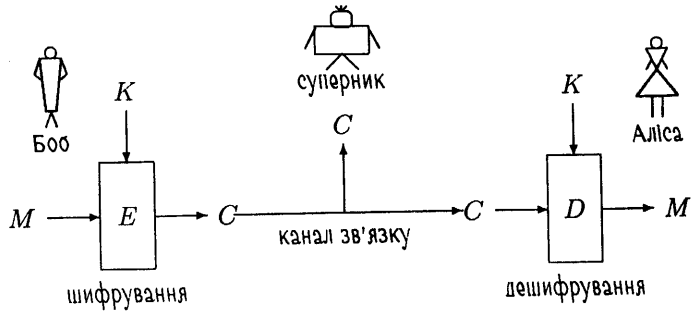
Шифр Цезаря. Давньоримський імператор Юлій Цезар (100–44 р. до н.е.) полюбляв зашифровувати свої таємні послання у спосіб, коли кожна буква заміщується деякою іншою, а саме тою, що знаходиться у алфавіті через три позиції. Стосовно української абетки це означає що, **а** міняється на **г**, **б** на **ґ**, **в** на **д** і т.д. Останні ж букви абетки **ь**, **ю**, та **я** зміщуються теж на три позиції *циклічно*, тобто переходять у **а**, **б** та **в**, відповідно. Для прикладу, слову **імперія** відповідає криптотекст **лптзулв**. Кажуть, що шифр Цезаря є *шифром зсуву* на 3 позиції.

Шифр частоколу. Алгоритм шифрування найліпше пояснити на прикладі. Щоб зашифрувати повідомлення **криптографія**, переписуємо його у вигляді "частоколу" $\text{к}^{\text{р}}\text{и}^{\text{п}}\text{т}^{\text{о}}\text{г}^{\text{р}}\text{а}^{\text{ф}}\text{і}^{\text{я}}$ і зчитуємо текст рядками, почавши з верхнього. В результаті отримуємо криптотекст **рпорфякитгаі**. "Висоту" частоколу можна вибирати. Щойно вона в нас була 2. Для висоти 3 ми на проміжному етапі мали б $\text{к}^{\text{р}}\text{и}^{\text{п}}\text{т}^{\text{о}}\text{г}^{\text{р}}\text{а}^{\text{ф}}\text{і}^{\text{я}}$, а остаточним результатом був би криптотекст **игірпорфякта**.

Довжина зсуву і висота частоколу у розглянутих шифрах є секретним *ключем*, який використовується як алгоритмом шифрування для перетворення відкритого тексту у криптотекст, так і алгоритмом дешифрування для зворотнього перетворення. Аліса та Боб домовляються про ключ завчасу (при особистій зустрічі, через надійного посередника тощо), зберігають його у таємниці і досить часто його міняють.

[1] Повідомлення записане назворот, тобто, його слід читати не зліва направо, а навпаки.

Ми описали найпростішу криптографічну ситуацію. Схематично вона зображена на малюнку 1. Літерами M та C на малюнку позначено відкритий текст та криптотекст, а літерою K — ключ. Літерами E та D позначені алгоритми шифрування та дешифрування відповідно. Цих позначень ми будемо дотримуватись і надалі.



Малюнок 1. Класична криптографічна схема.

Те, що криптотекст C є результатом застосування алгоритму E до відкритого тексту M та ключа K, записуватимемо як $C = E_K(M)$. Подібним чином, запис $M = D_K(C)$ означає, що M отримується із C та K за допомогою алгоритму D.

Аналізуючи надійність шифру, ми зобов'язані виходити із припущення, що суперник не лише здатен підслухати C, але й знає алгоритми E і D; і тільки ключ K йому невідомий.

Для знаходження M суперник може скористатися методом *повного перебору*. Тобто, суперник може перебирати всі можливі ключі K і обчислювати $D_K(C)$ доти, доки не натрапить на осмислений текст, який вірогідно і буде шуканим. Тому у надійної криптосистеми кількість можливих ключів мусить бути досить великою, щоб повний перебір не можна було зробити ні за який розумний час навіть з використанням швидкодіючої обчислювальної техніки.

Зауважимо, що знаходження ключа не є єдино можливим способом досягти успіху суперникові. Прикладом може служити *частотний метод*, описаний у пункті 2.2 цього розділу.

Якщо суперникові пощастило віднайти спосіб знаходження повідомлення за криптотекстом, то кажуть, що він *розкрив* шифр. *Криптографія* є мистецтвом створення шифрів, а *криптоаналіз* — їх розкриття. Тож респектабельніше називати суперника *криптоаналітиком*. Криптографія та криптоаналіз — дві складові *криптології*. Зрозуміло, що поділ дещо умовний, адже криптограф не може бути впевненим у надійності шифру без проведення його криптоаналізу. У ширшому трактуванні, завдання криптоаналізу не лише ламати шифри, тобто доводити їх ненадійність, але й навпаки, доводити у вигляді математичної теореми надійність шифру, попередньо означивши, який саме шифр слід вважати надійним.

Отже, перед криптоаналітиком стоїть завдання відновити повідомлення M. Згідно із традиційною термінологією, він проводить *атаку* на шифр. Так, метод повного перебору ключів ми будемо називати також *брутальною атакою*. Для розв'язування свого завдання криптоаналітик може мати різні передумови. Для останніх прийнята така класифікація.

Атака лише із криптотекстом. Суперник знає лише криптотекст $E_K(M)$. Досі саме така ситуація і малася на увазі. У гіршому випадку (кращому з погляду суперника), крім $E_K(M)$ відома ще певна кількість криптотекстів $E_K(M_1), \ldots, E_K(M_l)$, зашифрованих з використанням одного й того ж ключа.

Атака з відомим відкритим текстом. Крім $E_K(M)$ суперник знає як додаткові криптотексти $E_K(M_1), \ldots, E_K(M_l)$, так і відповідні їм відкриті тексти $M_1, \ldots, M_l$ (які, скажімо, пересилалися раніше і з тих чи інших причин вже не є таємними).

Атака з вибраним відкритим текстом. Суперник має доступ до "шифруючого устаткування" і спроможний отримати криптотексти $E_K(M_1), \ldots, E_K(M_l)$ для вибраних на власний розсуд відкритих текстів $M_1, \ldots, M_l$.[2]

Атака з вибраним криптотекстом. Суперник має доступ до "дешифруючого устаткування" і спроможний отримати відкриті тексти $D_K(M_1), \ldots, D_K(M_l)$ для вибраних на власний розсуд криптотекстів $C_1, \ldots, C_l$ (однак, як і у випадку попередньої атаки, неспроможний отримати безпосередньо таємний ключ).

Якщо атака певного виду призводить до розкриття шифру, то шифр є *вразливим* до неї; якщо ж ні, то шифр є *стійким* до такого виду атаки.

1.2. Термінологічні застереження. Як зазначалось, криптологія є наукою, яка поділяється на криптографію та криптоаналіз. Однак

[2]Ця атака відповідає мінімальним можливостям суперника у випадку криптосистем з відкритим ключем, яким присвячено розділ V.

досить часто термін криптографія вживається у загальнішому розумінні як синонім до криптології. *Захист інформації* є ширшою галуззю, що охоплює крім теоретичних основ також технічні засоби, юридичні аспекти тощо.

Тайнопис також є ширшим поняттям. Крім криптографічних, він допускає й такі способи збереження таємниці, при яких повідомлення ніяк не перетворюється, а приховується сам факт його існування чи пересилання. Прикладом може служити використання невидимого чорнила.

Вживання термінів *код* та *кодування* як синонімів до *шифру* та *шифрування* є не лише архаїчним, а часто й помилковим. Код — це усталене правило для заміни одиниць інформації (букв, слів, цілих фраз) певними символами. Наприклад, SOS є кодом прохання про допомогу, а згідно із ASCII кодом літера a кодується двійковою послідовністю 1100001. Коди, які вивчає математична *теорія кодування*, застосовуються з метою дещо протилежною до криптографічної. Повідомлення *шифрується* для того, щоб воно стало незрозумілим, а *кодується* для того, щоб бути зрозумілим навіть після часткового спотворення із-за природних перешкод у каналі зв'язку. Ці два терміни слід чітко розмежовувати тому, що на практиці одна й та ж інформація може підлягати обом діям — у типовій ситуації текст закодовують у двійкову послідовність, її шифрують, а отриманий криптотекст перед відправленням кодують за допомогою коду, який дозволить виправити помилки після передачі.

Як рівноправні терміни ми будемо вживати дієслова *шифрувати* і *криптувати* та віддієслівні іменники *шифрування* і *криптування*. Виключно справою смаку є вибір між словосполученнями *розкрити криптосистему* та *зламати криптосистему*. Нарешті, термін *криптотекст* має синонім *криптограма*, який щоправда зустрічається частіше у популярній, ніж у фаховій літературі.

1.3. Ретроспективні зауваження. Першим шифром, про який збереглася згадка, вважається шифр *Скитала*, який використовувався спартанцями для військових донесень у V ст. до н.е. Скиталою називався дерев'яний валик, на який щільно намотувалась стрічка пергаменту або шкіри. Повідомлення писалося рядками поздовж поверхні так, щоб у рядку на один звій припадала лише одна буква. Знята з валика стрічка містила незрозумілу послідовність букв, яку можна було прочитати лише знову намотавши стрічку на валик такого ж діаметру. Таким

чином, ключем у цьому шифрі слугував діаметр валика.

Криптоаналіз відкрили араби. Опис частотного методу є у їхніх писемних джерелах початку XV століття.

Шифри, які використовувались починаючи від античних часів і аж до середини XX століття, можна класифікувати як шифри *перестановки* або *заміни*. Скитала і шифр частоколу є прикладами шифрів перестановки — при шифруванні всі букви повідомлення зберігаються, лише розміщуються в іншому порядку. Шифр Цезаря є прикладом шифру заміни — кожна буква повідомлення замінюється деякою іншою. Шифрам перестановки та заміни присвячено наступний параграф цього розділу.

З сьогоднішніх позицій можна вважати, що аж до середини нашого століття у криптографії активно розвивались класичні методи. Так, знаменитий шифр часу II-ої світової війни *Enigma* можна трактувати як вдалу на тоді технічну реалізацію шифру Блеза де Віженера, французького криптографа XVI сторіччя. Все ж винахід телеграфу та радіо дав потужний поштовх дослідженням у царині криптології, адже при наявності у суперника відповідного обладнання підслуховування стало для нього зовсім необтяжливою справою. Шифр *одноразового блокноту* був винайдений у 1917 році інженером Гілбертом Вернамом з AT&T для застосування у телетайпній мережі. Цей шифр є *абсолютно надійним*, лише деякі обставини звужують сферу його застосування, як от необхідність мати безпечний канал для обміну довгим ключем.

Перша електронно-обчислювальна машина була сконструйована задля зламу шифру Enigma за участю видатного англійського математика Алана Тьюрінга. Поява обчислювальної техніки докорінно змінила критерії ефективності та надійності шифру. Шифр вважається *надійним в обчислювальному сенсі*, якщо його розкриття хоча в принципі й можливе, але навіть на найшвидшому комп'ютері вимагатиме нереального часу (роки, століття і т.д.), після завершення якого будь-яка таємниця стане неактуальною. Популярною криптосистемою, надійною саме в такому сенсі, є DES — стандарт шифрування даних прийнятий у Сполучених Штатах. Шифр одноразового блокноту та DES ми розглянемо нижче у § 3.

Початком сучасного етапу розвитку криптографії є 1976 рік. З'являються принципово нові криптографічні засоби, що дозволяє називати цей етап революційним. Давня задача збереження конфіденційності повідомлення знайшла нове елегантне розв'язання у концепції відкритого ключа. Кардинальна відмінність криптосистеми із відкри-

тим ключем (за іншою термінологією, асиметричної системи) від системи "дореволюційного зразка" (симетричної системи) полягає в тому, що у криптосистемах з відкритим ключем процедура шифрування стає загальнодоступною, але це не означає як у традиційних криптосистемах, що загальнодоступним є також дешифрування. Поняття ключа розбивається на дві частини: ключ відкритий, та ключ таємний. Загальновідомий відкритий ключ використовується для шифрування, але дешифрування може здійснити лише той, хто володіє таємним ключем. Криптосистеми з відкритим ключем є предметом розгляду у розділі V.

Озброєна новою концепцією, криптографія почала розв'язувати і нетрадиційні доволі софістичні завдання, як от підписування документів у електронній формі, жеребкування телефоном, поділ секрету між кількома особами тощо. Подібним задачам присвячено розділ VII.

ВПРАВИ

1.1. Користуючись шифром зсуву на 2 позиції, зашифрувати повідомлення **рятуйтесь**.

1.2. Розшифрувати криптотекст **мнзалмхэ**, отриманий за допомогою шифру зсуву на 31 позицію (пригадаємо, що в українській абетці 33 літери).

1.3. Розшифрувати криптотекст, отриманий за допомогою шифру зсуву із невідомим ключем: a) **бвсблбебвсб**; b) **мдодпдбдпчдмд**; c) **фхлфлтлчлг**; d) **тсжсусіьгос**.

1.4. Зашифрувати повідомлення **переховуватися** за допомогою шифру частоколу висоти a) 2; b) 3.

1.5. Розшифрувати криптотекст **нйеаінндо**, якщо відомо, що застосовано шифр частоколу невеликої висоти.

1.6. При шифруванні повідомлень записуванням їх назворот деякі повідомлення залишаються без зміни, як наприклад, **кіт втік** (пропуски між словами ігноруються). Тексти з такою властивістю називаються *паліндромами*. Скільки є паліндромів, які складаються з n букв української абетки (не обов'язково змістовних)?

ЛІТЕРАТУРА

Канонічним джерелом з історії криптографії є [108]. Цікаві зауваження ретроспективного характеру зроблено в оглядах [141, 131, 44]. Із доступних популярних видань можна порадити [23, 25]. Взаємозв'язки між криптографією і теорією кодування висвітлено в [52].

§ 2. Класичні методи: від античності до нового часу

У цьому параграфі ми розглянемо шифри *перестановки* та *підстановки*. Останні називають також шифрами *заміни*.

2.1. Шифри простої заміни перетворюють відкритий текст таким чином, що кожен символ замінюється на якийсь інший. При цьому однаковим символам у відкритому тексті відповідають однакові символи у криптотексті, а різним — різні. Ключем є табличка, що вказує в який саме символ переходить кожен символ відкритого тексту. Для прикладу, шифр Цезаря в українському алфавіті задається таким ключем:

а б в г ґ д е є ж з и і ї й к л м н о п р с т у ф х ц ч ш щ ь ю я
г ґ д е є ж з и і ї й к л м н о п р с т у ф х ц ч ш щ ь ю я а б в

При шифруванні кожна буква, яка зустрічається у повідомленні, шукається у верхньому рядку і замінюється відповідною буквою із нижнього рядка (пропуски між словами та розділові знаки ігноруються). Наприклад, **криптографія** перетворюється у **нуйтхсеугчкв**.

Алгоритм шифру Цезаря підставляє замість кожної букви алфавіту деяку букву того ж алфавіту. Могла б здійснюватись підстановка будь-яких інших символів, скажімо, ієрогліфів або ж піктограм танцюючих чоловічків як у випадку, описаному Артуром Конан Дойлом у однойменному оповіданні [22]. Має бути зрозумілим, що використання екзотичних символів у криптотексті хіба що надасть шифрові підстановки романтичного присмаку, але не зробить його надійнішим. Тому надалі ми без втрати загальності будемо вважати, що повідомлення та криптотекст пишуться в одному й тому ж алфавіті.

Зробивши це припущення, ми можемо підрахувати кількість всіх можливих ключів. Зробимо це для української абетки. Ключ — це табличка, верхній рядок якої містить всі букви у алфавітному порядку, а нижній — теж всі букви, але довільним чином перемішані. Отже питання полягає в тому, скількома способами можна розмістити всі букви абетки у нижньому рядку. Міркуємо так. Для першої позиції букву можна вибрати 33-ма способами. Після того як вона вибрана, для другої позиції букву можна вибрати вже 32-ма способами, для третьої — 31-м способом і т.д. Для передостанньої позиції вибір здійснюється 2-ма способами, остання ж буква визначена однозначно. Загальна кількість можливостей розмістити букви у всіх 33 позиціях дорівнює добутку $33 \cdot 32 \cdot 31 \cdot \ldots \cdot 2$. Таким чином, загальна кількість ключів є $33!$.

В загальному випадку, коли алфавіт налічує n символів, результатом такого ж підрахунку числа всіх ключів буде $n!$. Зауважимо, що серед цієї загальної кількості деякі ключі є непридатними для вжитку, як от тривіальний ключ, у якому нижній рядок збігається з верхнім.

Шифр зсуву є звуженням загального шифру заміни на сукупність лише n ключів, у яких нижній рядок є циклічним зсувом верхнього рядка. Ключ такого гатунку повністю визначається довжиною зсуву s. Можна вважати, що $0 \le s < n$, оскільки зсуви на s і на $s+n$ позицій дають однаковий результат.

Інші підкласи шифру простої заміни розглядаються у § II.3.

2.2. Частотний аналіз. Як нам відомо з попереднього пункту, шифр заміни над n-символьним алфавітом має $n!$ ключів. Для значень $n = 26, 33$ (латинський та український алфавіти) це число є дуже великим. Для його оцінки можна скористатися варіантом формули Стірлінга В.1, звідки для $n = 26$ отримуємо $n > 10^{26}$. Число справді велике — нагадаємо, що наша планета існує лише 10^9 років, а наступний льодовиковий період очікується через 14000 років, тобто $4,41504 \cdot 10^{11}$ секунд [137]. Це співставлення переконливо засвідчує безперспективність брутальної атаки на шифр заміни, однак цього недостатньо аби стверджувати, що він є надійним. Виявляється, успішний криптоаналіз можливий за допомогою *частотного методу*.

Частота символу у тексті дорівнює кількості його входжень у цей текст, поділеній на загальну кількість букв у тексті. Наприклад, частота букви **а** у тексті **купила␣мама␣коника** дорівнює $4/18 = 2/9$, а частота пропуску між словами у цьому ж тексті дорівнює $2/18 = 1/9$. Для кожної мови справджується такий емпіричний факт:

> *У досить довгих текстах кожна буква зустрічається із приблизно однаковою частотою, залежною від самої букви і незалежною від конкретного тексту.*

На підставі цього факту із кожним символом пов'язують деяке число, *частоту цього символу в мові*, з якою приблизно він зустрічається в кожному великому тексті цією мовою. Підрахунок частот символів у мові здійснюють на основі вибраного навмання середньостатистичного тексту. У додатку А наведено таблички частот для української, російської та англійської мов.

Припустимо, що перехоплено довгий криптотекст (або послідовність багатьох коротких), отриманий за допомогою шифру заміни. Частотним методом можна здійснити дешифрування, навіть не знаючи ключа. Для цього обчислюють частоти кожного символу в криптотексті і порівнюють отримані результати з табличкою частот для мови, якою написане повідомлення. Не слід сподіватися, що таким чином можна буде однозначно встановити ключ, але перебір це дозволить скоротити радикально. Наприклад, якщо при шифруванні не ігноруються пропуски між словами, то найпоширеніший символ у криптотексті поза сумнівом відповідає саме пропуску. А відтак стає відомою сукупність символів, що відповідають словам з одної букви (в українській мові це **а,б,в,є,ж,з,і,й,о,у,я**) та словам з двох букв (**це,не,на,до** та інші), що дозволяє ці символи розпізнати ціною справді невеликого перебору. Свою роль при частотному аналізі відіграє та обставина, що кожна мова володіє властивістю *надлишковості*, тобто текст можна поновити навіть коли частина його букв невідома.

Миттєвою є користь від частотного аналізу при розкритті шифру зсуву. Проілюструємо загальну ідею таким прикладом. Нехай нам належить розшифрувати криптотекст **пцпспофнпмплпибгпефрптмбвмєоп**, який був отриманий шифром зсуву, причому пропуски та розділові знаки ігнорувались. Підраховуємо частоти і зауважуємо, що найбільша, а саме 9/29, припадає на літеру **п**. Природньо припустити, що у відкритому тексті їй відповідає найпоширеніша в українській мові літера **о**. Це означало б, що довжина зсуву дорівнює 1. Виконуємо обернений зсув, тобто на одну позицію вліво, і справді отримуємо змістовне повідомлення, що **охоронумолокозаводупослаблено**.

Гомофонний шифр заміни був винайдений великим німецьким математиком Карлом Фрідріхом Гаусом [141]. Цей шифр грунтується на ідеї, яка робить підрахунок частот символів безперспективним. Кожна буква відкритого тексту замінюється не єдиним символом як у шифрі простої заміни, а будь-яким символом із декількох можливих. Наприклад, замість **а** може здійснюватись підстановка будь-якого із чисел **10,17,23,46,55**, а замість **б** — будь-якого із **12,71**. Головне, щоб замість різних букв завжди підставлялись різні символи — ця вимога забезпечує можливість дешифрування. Вибір одного з можливих варіантів щоразу робиться випадково. Якщо кількість варіантів для кожної букви пропорційна її частоті в мові, то всі символи у досить довгому криптотексті зустрічатимуться з приблизно однаковою частотою, що не дозволить пов'язати їх з якимись буквами відкритого тексту. Однак гомофонний шифр піддається ретельнішому і трудомісткішому різновиду частотного аналізу, який окрім частот окремих символів враховує

також частоти пар символів. Подібний аналіз дозволяє ламати ще один клас шифрів заміни, про що йдеться у наступному пункті.

Частотний аналіз може бути корисним і в інших ситуаціях. Наприклад, він дозволяє комп'ютерові без участі людини відрізнити осмислений текст від хаотичного набору символів. Завдяки цьому на машину можна перекласти здійснення брутальної атаки, тобто повного перебору ключів.

2.3. Поліграмні шифри. Послідовність кількох букв тексту називається *поліграмою*. Послідовність із двох букв називається *біграмою* (іноді *диграфом*), а із l букв — *l-грамою*. 3- і 4-грами називають відповідно три- і тетраграмами[3]. *Поліграмний шифр заміни* полягає у розбитті відкритого тексту на l-грами для деякого фіксованого числа l і заміні кожної з них на якийсь символ чи групу символів. Ключем є правило, за яким відбувається заміна. Якщо загальна кількість символів у тексті не ділиться націло на l, то остання група символів доповнюється до l-грами довільним наперед обумовленим способом.

Як приклад розглянемо *біграмний* (часом кажуть *диграфний*) шифр, який назвемо *шифром чотирьох квадратів,* хоча насправді він є відміною відомого шифру *Playfair* (початок XVI століття). Цей шифр застосовується до текстів латинкою. Точніше, ми нехтуємо літерою j, яка найрідше зустрічається в англомовних текстах, і працюємо з 25-літерним алфавітом[4]. Ключем є чотири квадрати розміру 5 на 5, кожен з яких сформований із всіх 25 букв, розташованих у довільному порядку. Зручно розмістити ці чотири квадрати так, щоб вони утворювали один великий квадрат як у прикладі на малюнку 2.

Перед шифруванням із повідомлення вилучаються всі розділові знаки, пропуски між словами, а також літера j, після чого повідомлення розбивається на біграми. Кожна біграма заміщується деякою іншою, яка визначається за таким правилом. Перша буква біграми, що підлягає заміщенню, відзначається у верхньому лівому квадраті, а нижня друга — у нижньому правому. Далі беруться дві букви, одна у верхньому правому, а друга у нижньому лівому квадратах, так, щоб разом з двома відзначеними буквами вони утворювали вершини прямокутника. Саме ці дві букви є біграмою, яка з'являється у криптотексті. Наприклад, при використанні ключа, зображеного на малюнку 2, біграма

[3]Терміни *монограма* і *пентаграма* випадають із цього ряду, бо мають інші значення.

[4]Та й до середніх віків буква j в латинській абетці була відсутня.

k	i	n	g	d	v	q	e	o	k
o	m	a	b	c	w	r	f	m	i
e	f	h	l	p	x	s	h	a	n
q	r	s	t	u	y	t	l	b	g
v	w	x	y	z	z	u	p	c	d

z	y	x	w	v	d	c	p	u	z
u	t	s	r	q	g	b	l	t	y
p	l	h	f	e	n	a	h	s	x
c	b	a	m	o	i	m	f	r	w
d	g	n	i	k	k	o	e	q	v

Малюнок 2. Приклад ключа для біграмного шифру чотирьох квадратів.

cr замінюється біграмою mo, а слово cryptography перетворюється у mopwtiomfxns.

Інші приклади поліграмних шифрів, які оперують l-грамами для довільного фіксованого l, є предметом розгляду в § II.3.

Зрозуміло, що для поліграмних шифрів при $l > 1$ підрахунок частот окремих букв алфавіту нічого не дає. Однак для $l = 2$ з успіхом застосовується аналіз частот біграм. У додатку А наведено найпоширені в українській мові біграми.

2.4. Поліалфавітні шифри можна потрактувати як такі шифри заміни, в яких позиція букви у відкритому тексті впливає на те, за яким саме правилом ця буква буде змінена. Ми розглянемо два класичні приклади.

Шифр Віженера. Відкритий текст і криптотекст записуються в одному й тому ж алфавіті. Для букв x та y цього алфавіту[5] означимо їх суму $x + y$ як результат циклічного зсуву букви x вправо у алфавіті на кількість позицій, що дорівнює номеру букви y в алфавіті. При цьому дотримуємося такої домовленості:

[5]Символами x та y ми позначаємо тут будь-які букви довільного алфавіту, а не лише латинські літери x та y.

нумерація букв алфавіту починається з нуля.

Наприклад, для української абетки маємо а+а = а, б+а = б, в+б = г, я+в = б. Ця операція природним чином задається так званою таблицею Віженера, яка зображена на малюнку 3.

	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я
а	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я
б	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а
в	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б
г	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в
ґ	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г
д	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ
е	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д
є	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е
ж	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є
з	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж
и	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з
і	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и
ї	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і
й	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї
к	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й
л	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к
м	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л
н	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м
о	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н
п	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о
р	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п
с	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р
т	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с
у	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т
ф	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у
х	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф
ц	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х
ч	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц
ш	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч
щ	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш
ь	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ
ю	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь
я	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю

Малюнок 3. Таблиця Віженера. Буква $x + y$ знаходиться на перехресті рядка, що відповідає букві x, і стовпчика, що відповідає букві y.

Шифр Віженера застосовують до повідомлення, записаного в рядок без пропусків між словами та розділових знаків. Ключем є слово у тому ж алфавіті. Якщо ключ коротший за повідомлення, то його записують багато разів підряд доки не вийде рядок такої ж довжини. Рядок з розмноженим ключем розміщують під рядком з повідомленням, і букви, що опинилися одна над другою, додають. В результаті отримують ще

один рядок тої ж довжини, який і є криптотекстом. Наприклад, шифрування наказу "**бороніть королівну від ворогів**" з ключем "**ключ**" відбувається так:

```
  БОРОНІТЬКОРОЛІВНУВІДВОРОГІВ
+ КЛЮЧКЛЮЧКЛЮЧКЛЮЧКЛЮЧКЛЮЧКЛЮ
  ЛАОЇЮЦРФШАОЇЩЦАІҐНЗЯМАОЇНЦА
```

Результатом шифрування є нижній рядок.

Як бачимо, на відміну від шифру простої заміни при використанні шифру Віженера однаковим буквам у відкритому тексті можуть відповідати різні букви у криптотексті. Ця обставина безперечно ускладнює частотний криптоаналіз. Шифр Віженера кілька століть вважався надійним, поки у 60-их роках минулого століття офіцер прусського війська Касискі не виявив, що цей шифр все ж піддається частотному методу. Скористаємось попереднім прикладом для пояснення основної ідеї. Шифр Віженера влаштований так, що при довжині ключа 4 кожна з чотирьох підпослідовностей відкритого тексту

```
Б␣␣␣Н␣␣␣К␣␣␣Л␣␣␣У␣␣␣В␣␣␣Г␣␣
␣О␣␣␣І␣␣␣О␣␣␣І␣␣␣В␣␣␣О␣␣␣І␣
␣␣Р␣␣␣Т␣␣␣Р␣␣␣В␣␣␣І␣␣␣Р␣␣␣В
␣␣␣О␣␣␣Ь␣␣␣О␣␣␣Н␣␣␣Д␣␣␣О␣␣␣
```

перетворюється відповідно до деякого шифру зсуву (на 14, 15, 31 і 27 позицій). За умови, що текст досить довгий, всі чотири довжини зсувів знаходяться стандартним підрахунком частот букв у відповідних підпослідовностях криптотексту. Однак як визначити із криптотексту, що застосовано ключ довжини саме 4? При прискіпливішому погляді на криптотекст зауважуємо у ньому однакові шматки — триграма **аої** зустрічається тричі, а біграма **ца** двічі. Природньо припустити, що це зумовлене не випадковістю, а тим, що у відкритий текст у відповідних місцях входить одна й та ж триграма та біграма (справді, у нашому випадку це **оро** та **ів**). Те, що дві однакові поліграми відкритого тексту проявились у криптотексті, означає, що відстань між ними є кратною довжині ключа. Відстані між різними входженнями триграми **аої** дорівнюють 8 і 12. Звідси висновок, що довжина ключа має ділити обидва ці числа, тобто вона дорівнює 1, або 2, або 4, — і нам лишається випробувати лише три можливості, щоб знайти, яка з них в дійсності має місце.

Зрозуміло, що описана метода може розраховувати на успіх завжди, коли довжина тексту співвідносно із довжиною ключа є великою. За цієї умови слід сподіватись, що текст міститиме чимало однакових бі-

грам та триграм, і частині з них відповідатимуть однакові біграми та триграми у криптотексті з тої причини, що відстань між ними пропорційна до довжини ключа.

Шифр з автоключем грунтується на ідеях Віженера і Кардано. Подібно до шифру Віженера, криптотекст отримують сумуванням відкритого тексту із послідовністю букв такої ж довжини. Однак тепер цю послідовність формують хитріше — спершу записують ключ, а справа до нього дописують початковий відрізок самого відкритого тексту. Якщо розглянути той же приклад, то шифрування відбуватиметься так:

```
  БОРОНІТЬКОРОЛІВНУВІДВОРОГІВ
+ КЛЮЧБОРОНІТЬКОРОЛІВНУВІДВОР
  ЛАОЇОЩЗЛЮЩЗЛЩЩТВДЙЙТХРЮУДЩТ
```

2.5. Шифрування блоками. Часто алгоритм шифрування буває призначений для перетворення послідовностей символів лише фіксованої довжини l. Коли ж потрібно застосувати його до більшого тексту, цей текст розбивають на *блоки* — групи по l символів, і кожен блок перетворюють окремо. Такі шифри називаються *блоковими з періодом* l. Якщо загальна кількість символів у тексті не ділиться націло на l, то остання група символів доповнюється до повного блоку довільним наперед обумовленим способом.

Прикладом блокового шифру з періодом l може слугувати будь-який поліграмний шифр, що здійснює заміну l-грам. Стосовно термінології зазначимо, що під поліграмою розуміють послідовність літер деякої природної мови, в той час як блок може складатися із довільних символів, скажімо, цифр.

Шифр Віженера з фіксованою довжиною ключа l теж можна розглядати як блоковий шифр з періодом l.

2.6. Шифри перестановки зберігають всі букви відкритого тексту, але розміщують їх у криптотексті в іншому порядку. Прикладами шифрів перестановки, які нам вже зустрічалися, є шифр частоколу і Скитала. Це представники широкого підкласу шифрів перестановки, які називаються шифрами *обходу*. У якості ще одного типового прикладу розглянемо *матричний шифр обходу*. Повідомлення записується рядками у вигляді прямокутної матриці. Криптотекст формується зчитуванням букв із матриці у зміненому порядку, а саме, стовпчиками. При цьому послідовність, у якій зчитуються стовпчики, визначається ключем. Було поширеним задання ключа у вигляді ключового слова,

що легко запам'ятовувалось. Порядок зчитування стовпчиків збігався з алфавітним порядком букв ключового слова. Приклад наведено на малюнку 4.

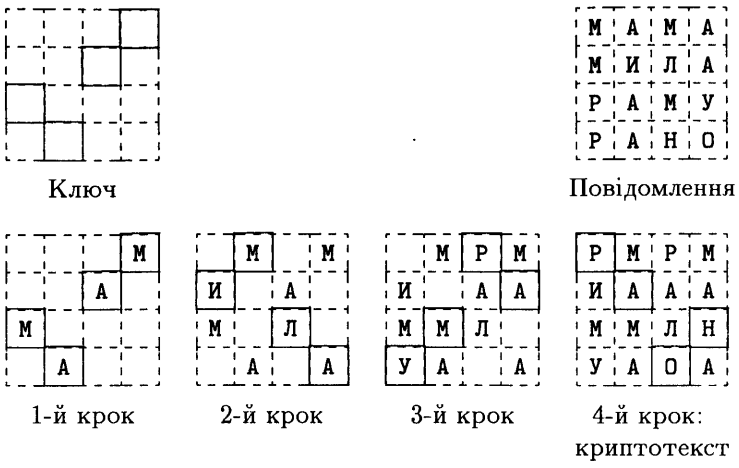
```
GARDEN      Повідомлення:
416235        DON'T PUT IT OFF TILL TOMORROW.
------
DONTPU      Ключове слово:
TITOFF        GARDEN
TILLTO      Криптотекст:
MORROW        OIIOTOLRPFTODTTMUFOWNTLR
```

Малюнок 4. Матричний шифр обходу.

Дещо відхиляючись від теми, необхідно зазначити, що в наш час вживання в якості ключа для будь-якого шифру зручних для запам'ятовування ключових слів, є досить ризикованим із-за *словникової атаки*. Найпростіший варіант цього методу криптоаналізу полягає в укладанні списку із, скажімо, 100000 найвживаніших ключових слів, включно із географічними назвами та екзотичними термінами. Рафінованіші версії включають лексику із вузько-фахових публікацій автора повідомлення, послідовності із двох-трьох складів китайської мови тощо.

У якості ще одного прикладу шифру перестановки наведемо доволі відомий у популярній математиці *шифр Кардано*. Це блоковий шифр з періодом $l = k^2$, де k парне число. Ключем є вирізаний з паперу в клітинку квадрат розміру k на k, що складається з k^2 клітинок, четверту частину яких, цебто $k^2/4$, прорізують. На малюнку 5 подано приклад для $k = 4$, причому квадрат розліновано пунктиром, а контури чотирьох прорізаних клітинок виділено суцільною лінією.

Нехай потрібно зашифрувати блок повідомлення, у якому k^2 літер. Криптотекст записують на клітчастому папері у квадраті k на k. Процедура займає чотири кроки. На першому кроці на аркуш, на якому буде писатись криптотекст, накладають ключ і вписують перші $k^2/4$ літер повідомлення у прорізані клітинки, починаючи з верхнього рядка. На другому кроці ключ повертають на 90 градусів за годинниковою стрілкою відносно центру квадрату і у прорізані клітинки вписують наступні $k^2/4$ літер повідомлення. Подібним чином виконують третій



Малюнок 5. Шифр Кардано.

і четвертий кроки — щоразу ключ повертають на 90 градусів і у нові позиції прорізаних клітинок вписують чергові $k^2/4$ літер повідомлення. Ключ має бути виготовлений у такий спосіб, щоб при повороті прорізані у ключі клітинки попали на вільні клітинки аркуша, і в жодному разі не наклалися на клітинки вже заповнені на попередніх кроках (див. вправу 2.22). В результаті після четвертого кроку всі k^2 літер блоку повідомлення виявляються розміщеними в деякому порядку у квадраті k на k. Зчитавши їх рядками, отримують криптотекст. На малюнку 5 показано процедуру перетворення повідомлення **мама мила раму рано** у криптотекст **рмрмиаааммлнуаоа**.

Шифр Кардано є шифром перестановки спеціального виду, у якому правило переставлення букв у блоці зручне для реалізації на папері за допомогою ножиць та олівця. Загальний шифр перестановки з періодом l переставляє l букв у довільному порядку, який визначається ключем. Ключ зручно задавати табличкою $\begin{pmatrix} 1 & 2 & \dots & l \\ i_1 & i_2 & \dots & i_l \end{pmatrix}$, яка показує, що перша буква блоку відкритого тексту займає позицію i_1 у відповідному блоці криптотексту, друга буква переміщується на позицію i_2, і т.д. Наприклад, при $l = 4$ шифр перестановки з ключем $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{pmatrix}$, перетворює відкритий текст **мамамиларамурано** у криптотекст **амамалимумаронар**. А зображений на малюнку 5 ключ для шифру Кардано можна задати табличкою $\begin{pmatrix} 1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12\,13\,14\,15\,16 \\ 4\,7\,9\,14\,2\,5\,11\,16\,3\,8\,10\,13\,1\,6\,12\,15 \end{pmatrix}$.

Міркуваннями, подібними до викладених у пункті 2.1, легко довести, що шифр перестановки з періодом l має $l!$ різних ключів. Якщо l є невеликим у порівнянні з довжиною тексту, шифр перестановки розкривається спеціальним чином організованим аналізом частот біграм. Як саме це відбувається, ми пояснимо на такому прикладі. Нехай маємо криптотекст

ІЦКАЗИВИМЯИЛКИНОЙРЕСРПІНЗМЕЛБОПИРПИИТИНИИВІСВИТАЛП

і відомо, що він отриманий шифром перестановки з періодом 5. Розіб'ємо криптотекст на блоки по 5 букв і запишемо їх один під другим, розташувавши текст у десяти рядках та п'яти колонках як показано на малюнку 6.

ІЦКАЗ		ЗАКЦІ
ИВИМЯ	← Криптотекст	ЯМИВИ
ИЛКИН	Відкритий текст →	НИКЛИ
ОЙРЕС		СЕРЙО
РПІНЗ		ЗНІПР
МЕЛБО		ОБЛЕМ
ПИРПИ		ИПРИП
ИТИНИ		ИНИТИ
ИВІСВ		ВСІВИ
ИТАЛП		ПЛАТИ
12345	← Номери стовпчиків →	54321

Малюнок 6. Криптоаналіз шифру перестановки.

Зауважимо тепер, що дешифрування полягає у переставленні стовпчиків у належному порядку. Порядок цей, не знаючи ключа, можна встановити такими міркуваннями. На першій і третій позиціях другого рядка стоїть літера **и**. Це означає, що перший і третій стовпчики не можуть стояти поруч, інакше ми мали б у відкритому тексті біграму **ии**,

яка ніколи в українській мові не зустрічається[6], навіть якщо з тексту вилучено пропуски між словами. Зауважуємо також подвійні входження літери **и** у третьому та сьомому рядках і потрійне входження у восьмому рядку. Беручи до уваги позиції букви **и** у цих рядках, приходимо до висновку, що не можуть знаходитись поруч 1-й і 4-й, 2-й і 5-й стовпчики, а також будь-які два із 1-го, 3-го та 5-го стовпчиків. Неважко пересвідчитись, що ці вимоги задовольняють лише два розташування стовпчиків — (1,2,3,4,5) та (5,4,3,2,1). Оскільки перше розташування відповідає нашому криптотекстові, то для розташування стовпчиків у відкритому тексті залишається єдина можливість — (5,4,3,2,1), що відповідає ключеві $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 4 & 3 & 2 & 1 \end{pmatrix}$. Здійснивши обернену перестановку, прочитуємо розшифроване повідомлення по рядках: **З АКЦІЯМИ ВИНИКЛИ СЕРЙОЗНІ ПРОБЛЕМИ. ПРИПИНИТИ ВСІ ВИПЛАТИ!**

У нашому криптоаналізі ми виходили з апріорно заданої інформації, що шифрування здійснювалось блоками довжини $l = 5$. Якби цього не було відомо, належало б подібний аналіз проводити почергово для значень $l = 2, 3, 4, \ldots$ аж поки б не було досягнуто успіху. Для максимально наочної ілюстрації загального принципу приклад був підібраний так, що ключ було визначено однозначно на підставі єдиного факту — біграма **ии** зустрічається в українській мові з малою частотою (насправді нульовою). В загальному випадку беруть до уваги й інші малоймовірні буквосполучення, і в результаті отримують систему обмежень на ключ, яка як правило дозволяє суттєво скоротити перебір.

На завершення зробимо ще одне просте зауваження, корисне при атаці з відомим відкритим текстом. У цьому випадку легко визначити, що використовується шифр саме перестановки — досить пересвідчитись, що кожна буква зустрічається в повідомленні та криптотексті з однаковою частотою.

ВПРАВИ

2.1. a) Зашифрувати слово `cryptography` за допомогою шифру заміни з ключем

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
b a d c f e h g j i l k n m p o r q t s v u x w z y
```

b) Дешифрувати криптотекст `vmjufqtjsz`, отриманий за використанням того ж ключа.

[6] Єдиним винятком є слово *цилиспит* — назва мінералу, запозичена з мови одного із малочисельних народів Півночі *(примітка редактора)*.

2.2. a) Довести, що послідовне застосування шифру простої заміни двічі, один раз з ключем K_1, а другий раз з ключем K_2, еквівалентне одноразовому застосуванню шифру заміни з деяким ключем K_3.

b) Довести, що криптотекст, отриманий за допомогою шифру заміни з ключем K, можна дешифрувати, застосувавши шифр заміни з деяким ключем K'.

c) Нехай в позначеннях попереднього пункту $K' = K$, тобто дешифрування можна здійснити, застосувавши шифр з тим же ключем ще раз. Припустимо також, що при шифруванні з ключем K жоден символ алфавіту не залишається незмінним. Скільки ключів K для шифру заміни у 26-символьному алфавіті мають такі дві властивості?

2.3. Для букви a у n-символьному алфавіті та цілого числа s, означимо їх суму $a + s$ як результат циклічного зсуву букви a на s позицій у алфавіті, вправо якщо $s > 0$ і вліво інакше. Наприклад, для латинського алфавіту $\mathtt{a} + 3 = \mathtt{d}$, $\mathtt{b} + (-3) = \mathtt{y}$. Довести співвідношення a) $a + (s + n) = a + s$; b) $(a + s_1) + s_2 = a + (s_1 + s_2)$; c) $(a + s_1) + s_2 = (a + s_2) + s_1$ для будь-якої букви a та чисел s, s_1, s_2.

2.4. a) Довести, що послідовне застосування шифру зсуву двічі, один раз з ключем s_1, а другий раз з ключем s_2, еквівалентне одноразовому застосуванню шифру зсуву з ключем $s_1 + s_2$;

b) Довести, що криптотекст, отриманий за допомогою шифру зсуву з ключем s, $0 \le s < n$, можна дешифрувати, застосувавши шифр зсуву з ключем $n - s$.

2.5. Порахувати частоти всіх символів у тексті `мама␣мила␣раму`.

2.6. В потоці зашифрованих донесень від інформатора з Маріїнського палацу домінує буква н. Припустивши, що використовується шифр зсуву і пропуски між словами ігноруються, знайти за допомогою частотного аналізу ключ і розшифрувати повідомлення

`опдрснкнмярстомэйнлопнвнкнчдмннкдкщйяспдсщнвн`.

2.7. a) Використовуючи шифр чотирьох квадратів із пункту 2.3 з ключем як на малюнку 2, зашифрувати слово `university`.

b) Дешифрувати криптотекст `sknromra`, отриманий за допомогою того ж шифру з тим же ключем.

2.8. a) Скільки різних k-грам є у n-символьному алфавіті?

b) Скільки різних біграм є у алфавіті з 25 букв?

2.9. a) Скільки різних ключів має шифр чотирьох квадратів, описаний у пункті 2.3?

b) Вказати два різні ключі для шифру чотирьох квадратів, шифрування з якими завжди дає однакові результати.

c) Скільки різних ключів має загальний біграмний шифр для 25-символьного алфавіту, який кожну біграму відкритого тексту переводить у

біграму криптотексту в тому ж алфавіті (зайве зазначати, що різні біграми переводяться у різні)?

d) Скільки з них можуть бути реалізовані як ключі для шифру чотирьох квадратів?

2.10. a) Розглянувши операцію додавання букв алфавіту, введену в пункті 2.4, довести, що для будь-яких букв x,y, та z виконуються рівності $(x+y)+z = x+(y+z)$ (асоціативність) та $x+y = y+x$ (комутативність).

b) Позначимо через X та Y довільні два слова однакової довжини. Довести, що шифрування слова X за допомогою шифру Віженера з ключем Y дасть той же результат, що й шифрування слова Y з ключем X.

2.11. a) Зашифрувати повідомлення **білі мухи налетіли**, використавши шифр Віженера з ключем **зима**.

b) Розшифрувати криптотекст **ьччжпчьишисаєяйпявааьяч**, отриманий за допомогою шифру Віженера з тим же ключем.

2.12. Показати, що шифр зсуву є частковим випадком шифру Віженера. Який ключ у шифрі Віженера над українським алфавітом слід взяти, щоб отримати шифр Цезаря?

2.13. a) Довести, що послідовне застосування шифру Віженера двічі, один раз з ключем K_1 і другий раз з ключем K_2 тої ж довжини, еквівалентне одноразовому застосуванню шифру з ключем K_3, де K_3 можна отримати шифруванням слова K_1 з ключем K_2.

b) Довести, що послідовне застосування шифру Віженера двічі, один раз з ключем K_1 довжини l_1 і другий раз з ключем K_2 довжини l_2, еквівалентне одноразовому застосуванню шифру з деяким ключем K_3 довжини l_3, де l_3 є найменшим спільним кратним чисел l_1 та l_2.

2.14. Вибрати для шифру Віженера довільний ключ довжини a) 3, b) 6 і зашифрувати повідомлення **бороніть королівну від ворогів**. У отриманому криптотексті знайти однакові триграми або біграми і порахувати, на якій відстані одна від одної вони знаходяться.

2.15. (Beaufort) Для букви x алфавіту через $-x$ позначимо *протилежну* їй букву алфавіту, а саме таку, що сума номерів обох букв дорівнює кількості букв у алфавіті. Нагадаємо, що нумерація в алфавіті починається з 0. Покладемо також, що перша буква є протилежною сама до себе. При криптуванні за допомогою шифру Віженера кожна буква c криптотексту отримується із відповідних букв m і k повідомлення і ключа за формулою $c = m + k$, де операція додавання описана на стор. 23. Розглянемо модифікацію шифру, при якій криптування здійснюється за формулою $c = (-m) + k$.

a) Зашифрувати повідомлення **білі мухи налетіли** за допомогою ключа **зима**.

b) Розшифрувати криптотекст **етишфхвлчишізшюяюшен**, отриманий за допомогою того ж ключа.

c) Довести, що у якості алгоритму декриптування для цієї модифікації шифру Віженера можна взяти просто алгоритм криптування. Шифр з такою властивістю називається *інволютивним*.

2.16. a) Зашифрувати повідомлення **білі мухи налетіли**, за допомогою шифру з автоключем, взявши як ключ слово **зима**.

b) Розшифрувати криптотекст **ьччжюбхощпшнтвпсккпікш**, отриманий за допомогою того ж шифру з тим же ключем.

2.17. Зламати таку версію шифру з автоключем. Якщо ключ має довжину l, то відкритий текст розбивають на блоки довжини l кожен, які послідовно перетворюють у блоки криптотексту наступним чином. Перший блок криптотексту є сумою першого блоку відкритого тексту і ключа. Кожен наступний блок криптотексту отримують сумуванням відповідного блоку відкритого тексту із попереднім блоком криптотексту. (Не знаючи ключа, за криптотекстом можна відновити відкритий текст за винятком перших l літер.)

2.18. Довести, що послідовне застосування до досить довгого тексту двох блокових шифрів, одного з періодом l_1, а другого з періодом l_2, призводить до шифрування блоками довжини l, де l є найменшим спільним кратним чисел l_1 і l_2.

2.19. a) Зашифрувати за допомогою матричного шифру обходу з ключем, зображеним на малюнку 4, повідомлення **you must strike while the iron is hot**.

b) Дешифрувати криптотекст **niuhngdsoneuahwtterndnne**, отриманий з допомогою того ж шифру з тим же ключем.

2.20. a) Зашифрувати за допомогою шифру Кардано з ключем, зображеним на малюнку 5, слово **недопереповнення**.

b) Дешифрувати криптотекст **ьетуркнеітсисвйи**, отриманий тим же шифром з тим же ключем.

2.21. Виготовити ключ 6 на 6 для шифру Кардано і зашифрувати з його допомогою довільний текст із 36 букв.

2.22. Скільки є для шифру Кардано ключів розміру a) 4 на 4; b) 6 на 6; c) k на k, для парного k?

2.23. a) Зашифрувати за допомогою загального шифру перестановки з періодом 6 і ключем $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 2 & 6 & 3 & 4 & 1 \end{pmatrix}$ слово **криптоаналіз**.

b) Розшифрувати криптотекст **оеекпреиолтп**, отриманий з допомогою того ж шифру з тим же ключем.

2.24. Перехоплено три криптотексти: **саннаа**, **бббаао** і **укаадк**. Як стало відомо, перші два відповідають повідомленням **ананас** і **баобаб**. Розшифрувати третій криптотекст.

2.25. a) Довести, що послідовне застосування шифру перестановки двічі, один раз з ключем K_1 і другий раз з ключем K_2, еквівалентне одноразовому застосуванню шифру з деяким ключем K_3 (період фіксований).

b) Довести, що криптотекст, отриманий за допомогою шифру перестановки з ключем K, можна дешифрувати, застосувавши той же шифр з деяким ключем K'.

c) Нехай у позначеннях попереднього пункту $K' = K$, тобто дешифрування можна здійснити, застосувавши шифр з тим же ключем ще раз. Припустимо також, що при шифруванні з ключем K кожна буква переходить на нове місце. Скільки ключів K для шифру перестановки з періодом l мають такі дві властивості?

2.26. a) Нехай A — шифр перестановки, а B — шифр заміни. Довести, що застосування спочатку шифру A, а потім B, еквівалентне застосуванню спочатку шифру B, а потім A з тими ж ключами. (В такому випадку кажуть, що шифри A і B *комутують*.)

b) Навести приклад двох шифрів перестановки, які не комутують.

c) Навести приклад двох шифрів заміни, які не комутують.

2.27. Довести, що послідовне застосування до досить довгого повідомлення двох шифрів перестановки з періодами l_1 та l_2, першого з ключем K_1 а другого з ключем K_2, еквівалентне одноразовому застосуванню шифру перестановки з деяким ключем K_3 та періодом l_3, що є найменшим спільним кратним чисел l_1 і l_2.

2.28. Використовується шифр перестановки з періодом 5. Визначити ключ і розшифрувати криптотекст

ничаэ нiюон куртс сюеiц терке ивйин рпбiр иниви
нненк ьдубi зiокя зорга щинзи сьтеу рохоя мецно

ЛІТЕРАТУРА

Авторитетними джерелами з елементарної криптографії є [112] та [143]. Неперевершений у світовій літературі опис частотного аналізу належить Едгару По [49]. Поняття надлишковості мови введене в [60]. Шифр чотирьох квадратів у пункті 2.3 описано в [110]. Популярний виклад шифру Кардано міститься в [48, Розділ VI].

§ 3. Дві пропозиції XX століття

Цей параграф значною мірою присвячено двом конкретним шифрам — *одноразового блокноту* і *DES*, та пов'язаним із ними концепціям. Важливо зазначити, що ці два шифри ілюструють два різні поняття

надійності. Шифр одноразового блокноту є надійним *у теоретико-інформаційному сенсі*, а DES ось уже двадцять років вважається надійним *в обчислювальному сенсі*.

3.1. Подання тексту у цифровій формі. Як ми бачили, у період класичної криптографії як правило не виникало потреби записувати відкритий текст та криптотекст якось інакше, ніж у звичайній абетці. Завдяки цьому криптограф-практик не потребував для роботи нічого, крім письмового приладдя свого часу, чого було достатньо і для шифрування, і для пересилання повідомлення. Але як тільки ми забажаємо скористатися модерними засобами зв'язку для передачі повідомлення, або доручити шифрування комп'ютерові, то виявимо, що у технічному відношенні традиційний текст не є найзручнішою формою для перетворення та передачі інформації.

З цього погляду вигіднішим є подання інформації у *цифровій формі*. Ідея є зовсім простою — кожен символ тексту заміняємо його номером у алфавіті. Нагадаємо, що нумерацію ми домовились починати з 0. Для прикладу, слово **банан** буде подане як **01 00 17 00 17**. Кожна літера представлена своїм номером, записаним двома цифрами, перша з яких може бути нулем. При потребі в алфавіт можна включити окрім букв також знаки пунктуації, пропуск, цифри тощо.

Номери букв ми можемо записувати не в десятковій системі числення, а у двійковій. Для того ж слова **банан** матимемо запис **000001 000000 010001 000000 010001**, де кожен блок із шести цифр є номером відповідної букви у двійковому записі[7]. Таку форму подання тексту називатимемо *двійковою*.

Таким чином, довільний текст можна записати у двійковій формі, використовуючи всього лише два символи — **0** та **1**. Ці два символи називаються *бітами*. Будь-яку послідовність бітів називають *двійковим словом*.

3.2. Шифр одноразового блокноту був винайдений у 1917 році Гілбертом Вернамом. Він використовує операцію додавання бітів за модулем 2, яку ми розглянемо перед тим як описати сам шифр. Операція позначається символом $\oplus$ і задається так:

$$0 \oplus 0 = 0, \quad 0 \oplus 1 = 1, \quad 1 \oplus 0 = 1, \quad 1 \oplus 1 = 0.$$

[7] Ми потребуємо не менше шести цифр, бо п'ятьма можна записати щонайбільше 32 числа, в той час як українська абетка налічує 33 літери.

Поширимо цю операцію на двійкові слова однакової довжини домовившись, що додавання таких слів здійснюється побітово[8]. Наприклад,

$$\begin{array}{l} \texttt{00000100000001000100000010001} \\ \oplus\texttt{00110111010110001001100011101 0} \\ \hline \texttt{00110011010111001101100010101 1} \end{array}$$

Для двійкових слів X і Y однакової довжини результат їх побітового додавання позначатимемо як $X \oplus Y$. Легко перевірити рівності $X \oplus Y = Y \oplus X$ та $(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z)$, що справджуються для будь-яких двійкових слів X, Y і Z однакової довжини. Для фіксованого k позначимо через 0 двійкове слово $\texttt{000...00}$, що складається із k нулів. Очевидно, що для будь-якого двійкового слова X довжини k виконуються рівності $X \oplus 0 = X$ і $X \oplus X = 0$.

Перейдемо тепер до опису шифру. Перед шифруванням повідомлення M записують у двійковій формі. Ключем K служить довільне двійкове слово однакової з M довжини. Криптотекст C отримують побітовим додаванням повідомлення і ключа, тобто $C = M \oplus K$.

Для прикладу, нехай ми хочемо зашифрувати слово **банан**. У попередньому пункті ми подали його у двійковій формі:

$$M = \texttt{00000100000001000100000010001}.$$

В якості ключа виберемо

$$K = \texttt{00110111010110001001100011101 0}.$$

Сумування цих двох двійкових послідовностей вже проведене нами вище. Отож маємо криптотекст

$$C = \texttt{00110011010111001101100010101 1}.$$

Дешифрування у шифрі одноразового блокноту збігається із шифруванням — щоб отримати вихідне повідомлення M, слід додати до криптотексту C той же ключ K. Це легко обгрунтувати: оскільки $C = M \oplus K$, то

$$C \oplus K = (M \oplus K) \oplus K = M \oplus (K \oplus K) = M \oplus 0 = M.$$

Шифр одноразового блокноту є *абсолютно надійним* або, як ще кажуть, *надійним у теоретико-інформаційному сенсі*. Якщо суперник не знає ключа K, то з підслуханого криптотексту C він *зовсім нічого* не може довідатись про повідомлення M. Справді, двійкове слово C могло би бути криптотекстом для *будь-якого* повідомлення M', якби шифрування здійснювалось з деяким іншим ключем K', а саме $K' = M' \oplus C$, в той час як для суперника всі ключі однаково вірогідні. Наприклад,

[8] Не слід плутати цю операцію із звичайним додаванням чисел у двійковій системі. Зокрема, при побітовому додаванні не виникає ніяких переносів у наступний розряд.

за допомогою деякого ключа K ми щойно перетворили повідомлення **банан** у криптотекст

$$C = \texttt{00110011010111001101100010101 1}.$$

Але такий же криптотекст ми отримали б, якби зашифрували повідомлення **груша** з ключем

$$K' = \texttt{00111110000110010000010010101 1}.$$

Назва шифру походить від того, що агент, який здійснював шифрування вручну, отримував свої копії ключів записаними у блокноті. Як тільки ключ використовувався, сторінка з ним знищувалась. Зрозуміло, що шифр просто реалізується і технічними засобами. Кажуть, що саме шифр одноразового блокноту використовувався для захисту від підслуховування встановленої під час холодної війни гарячої лінії зв'язку між Москвою і Вашингтоном.

Однак обмеженість сфери застосувань шифру очевидна, оскільки він вимагає ключа такої ж довжини як саме повідомлення. З цією обставиною пов'язані дві проблеми. Перша полягає в генеруванні довгої послідовності випадкових бітів і буде розглянута нами у розділі VI. Другою проблемою є необхідність у надійному каналі для регулярного обміну довгим ключем (на зразок дипломатичної пошти). У більшості ситуацій такого каналу або взагалі нема, або ж він не є достатньо швидким.

На завершення зауважимо, що достеменно такими ж перевагами та недоліками володітиме шифр Віженера, якщо у ньому брати ключ тої ж довжини, що й повідомлення.

3.3. Кількаразове шифрування. Алгоритм, який полягає у шифруванні повідомлення за допомогою одного шифру, а потім застосуванні до отриманого криптотексту ще одного шифру називається *композицією* або *добутком* цих двох шифрів.

Компонування шифрів (утворення їх композиції) часто використовувалось у ранній криптографії. Було зауважено, скажімо, що композиція матричних шифрів обходу призводить до перестановки, яку набагато важче задати саму по собі (вправа 3.5). Корисним є також спостереження, що в результаті компонування двох блокових шифрів із взаємно простими періодами період збільшується (вправа 2.18).

У роки І-ої світової війни з'явились *подрібнюючі системи*, чи не найвідомішим представником яких є шифр *ADFGVX*, що застосовувався німецькою армією. ADFGVX є композицією двох шифрів — підстановки і перестановки. Спочатку кожен символ повідомлення, який може

бути латинською літерою або десятковою цифрою, замінюється парою символів A, D, F, G, V, або X, згідно із табличкою розміру 6 на 6, як це показано на малюнку 7. Отриманий проміжний криптотекст знову зашифровується, цього разу за допомогою матричного шифру обходу.

	ADFGVX
A	C08XF4
D	MK3AZ9
F	NWL0JD
G	5SIYHU
V	P1VB6R
X	EQ7T2G

Таблиця білітеральної підстановки-подрібнення.

Повідомлення:
D O N'T P U T I T O F F T I L L T O M O R R O W
Проміжний криптотекст:
FXADFAXGVAGXXGGFXGADAVAVXGGFFFFFXGADDAADVXVXADFD

GARDEN
416235
FXADFA
XGVAGX
XGGFXG
ADAVAV
XGGFFF
FFXGAD
DAADVX
VXADFD

Матриця перестановки. Запис проміжного криптотексту рядками і зчитування колонками в порядку, визначеному ключовим словом GARDEN.

Криптотекст:
XGGDG FAXDA FVFGD DFGXA
FAVFF XXAXF DVAXG VFDXD
AVGAG XAA

Малюнок 7. Шифр ADFGVX.

У 20-их роках нашого століття були винайдені *роторні* шифрувальні пристрої, які вдосконалювались упродовж наступних десятиліть і інтенсивно використовувались під час II-ої світової війни. Прикладом може служити відомий німецький шифр Enigma. Роторні системи реалізовували багатократну композицію шифрів Віженера, що давало шифр з дуже великим періодом (див. вправу 2.13).

Частковим випадком композиції двох шифрів є послідовне застосування двічі одного й того ж шифру. Інтуїтивно правдоподібним виглядає припущення, що подвійне шифрування збільшує надійність. Втім, наведемо кілька прикладів, що мають застерегти від поквапливих висновків з цього приводу — питання потребує акуратного розгляду у кожному конкретному випадку.

Спершу припустимо, що обидва рази шифрування здійснюється з одним і тим же ключем. Зрозуміло, що такий спосіб зовсім не ускладнює брутальну атаку — потрібно перебрати ту ж кількість ключів. Більше того, є так звані *інволютивні* криптосистеми, як от шифр одноразового блокноту, у яких процедура шифрування збігається з процедурою дешифрування (див. також вправу 2.15). Для таких систем подвійне шифрування залишає повідомлення незміненим!

Перспективнішим видається подвійне шифрування з незалежним вибором ключа кожного разу. Тобто, за допомогою алгоритму шифрування E повідомлення M перетворюється у криптотекст $C = E_{K_2}(E_{K_1}(M))$, де два ключі K_1 та K_2 вибираються незалежно один від одного. Дешифрування не складає труднощів: $M = D_{K_1}(D_{K_2}(C))$ (належить звернути увагу на зміну порядку, в якому застосовуються ключі).

Припустимо, що ключем служить двійкове слово довжини n. В цьому разі всіх можливих ключів є 2^n, а всеможливих пар ключів є 2^{2n}. Однак це не означає, що ламання двократного шифру перебором займе 2^{2n} кроків замість 2^n. Є спосіб оптимізації перебірної процедури, так звана *зустрічна атака*, що займає кількість кроків пропорційну до 2^n, але взамін вимагає машинної пам'яті такого ж порядку. Це атака з відомим відкритим текстом, для проведення якої потрібно знати якусь пару з повідомлення M та відповідного йому криптотексту C. Зустрічна атака проходить таким чином. Маючи M і C, укладають дві таблиці. В першу поміщають значення $E_K(M)$ для всіх можливих ключів K, а у другу — значення $D_K(C)$ теж для всіх можливих ключів K. Обидві таблички впорядковують згідно з алфавітом, що використовується[9], і шукають в них однакові пари $E_{K_1}(M) = D_{K_2}(C)$ — слід сподіватися таких виявиться не надто багато. Серед знайдених пар ключів (K_1, K_2) виявиться та, яка справді була використана при подвійному шифруванні.

[9]Про методи сортування дивись [7, 62].

Для деяких шифрів їх кількаразове застосування не збільшує надійності з тої причини, що для таких шифрів подвійне шифрування з двома ключами K_1 і K_2 дає той же результат, що й одноразове шифрування з деяким ключем K_3, тобто, для будь-якого повідомлення M виконується рівність $E_{K_2}(E_{K_1}(M)) = E_{K_3}(M)$. Кажуть, що такі шифри *утворюють групу*. Прикладом є шифр одноразового блокноту — легко перевірити, що для нього $K_3 = K_1 \oplus K_2$. Іншими прикладами можуть служити шифр простої заміни, шифри зсуву, Віженера та перестановки (див. вправи 2.2, 2.4, 2.13, 2.25 і 2.27).

Розробники шифрів намагаються проектувати шифр так, щоб він не утворював групи. З успішним прикладом ми ознайомимось у наступному пункті, присвяченому алгоритмові DES, який здійснює 16-кратне повторення одного й того ж циклу перетворення інформації.

3.4. DES. *Стандарт шифрування даних* (англійською Data Encryption Standard — DES) був розроблений у 70-х роках фахівцями з IBM і у 1976 році був прийнятий через NBS та ANSI у якості федерального стандарту Сполучених Штатів для захисту комерційної та урядової інформації, не пов'язаної з національною безпекою. Цим актом увінчався цілий етап у розвитку криптографії — майже одночасно з'явилися принципово нові криптосистеми з відкритим ключем, яким присвячено більшість подальших розділів цієї книжки.

Спробуємо простежити уявний шлях від шифру одноразового блокноту до криптосистеми DES. Для цього нагадаємо дві основні характеристики першого. З одного боку, шифр одноразового блокноту є абсолютно надійним, а з іншого, він далеко не завжди є практичним через велику довжину ключа. Тому перший крок є таким — довжина ключа має бути фіксованою, а шифрування має відбуватися блоками. Як і шифр одноразового блокноту, DES оперує з інформацією поданою у двійковій формі, а довжина блоку і довжина ключа вибрані рівними 64. Іншими словами, двійкове повідомлення M розбивають на блоки по 64 біти і шифрують кожен блок окремо, використовуючи один і той же двійковий ключ K довжини 64. Таким чином, повідомлення $M = M_1M_2M_3\ldots$ перетворюється у криптотекст $C = C_1C_2C_3\ldots$, де $C_i = E_K(M_i)$. У стандарті DES кожен блок криптотексту C_i також є двійковою послідовністю довжини 64.

Звернемося до питання вибору алгоритму E. Апріорно він повинен задовільняти три умови:

можливість дешифрування: для будь-якого ключа K різним блокам повідомлення M' і M'' відповідають різні блоки криптотексту $E_K(M')$ і $E_K(M'')$, інакше кажучи, алгоритм E_K з будь-яким ключем здійснює перестановку двійкових послідовностей довжини 64;

ефективність: і шифрування, і дешифрування відбуваються швидко;

надійність: якщо ключ невідомий, то немає способу розкриття шифру.

Останній пункт потребує принципового уточнення. Зрозуміло, що як би вдало алгоритм E не був спроектований, надійність, що давалася шифром одноразового блокноту, вже буде недосяжною. Недосяжною з тої причини, що будь-який блоковий шифр можна зламати за допомогою брутальної атаки, яка у нашому випадку полягає в переборі всіх двійкових ключів K довжини 64. Про яку ж надійність йдеться? Зауважимо, що всього двійкових ключів довжини 64 є 2^{64}. Елементарні обчислення показують, що комп'ютер із тактовою частотою 100 MHz перебиратиме всі можливі ключі $2^{64}/10^8$ секунд, тобто довше ніж 5800 років. Звичайно, повний перебір необхідний лише у найгіршому випадку. У середньому, потрібний ключ буде знайдено вдвічі швидше — приблизно за 2900 років — час, за який таємниця втратить актуальність. Якщо для шифру невідомо методу його ламання упродовж більш реалістичного терміну, то такий шифр вважають *надійним в обчислювальному сенсі*.

Якщо ми хочемо отримати криптосистему надійну у такому новому розумінні, зразу відмовимось від наївної спроби вдосконалити шифр одноразового блокноту і покласти $E_K(M_i) = M_i \oplus K$. Таке шифрування є, що приємно, дуже швидке, але, що неприйнятно, вкрай ненадійне. Для розкриття досить просумувати за модулем 2 криптотекст із самим собою, але зсунутим на 64 позиції. Із рівності $K \oplus K = 0$ випливає, що результат буде тим же, що й при сумуванні повідомлення із самим собою зсунутим на ту ж кількість позицій. Із цієї інформації, яка вже ніяк не залежить від ключа, неважко отримати саме повідомлення, запорукою чого є феномен надлишковості будь-якої природної мови.

Розробники стандарту DES пропонують наступну конструкцію (подальший опис алгоритму спрощено задля акцентування найважливіших фаз його роботи).

Алгоритм E приймає на вхід двійковий блок M довжини 64 та використовує ключ K, з якого виділяються 16 частин $K_1, \ldots, K_{16}$ по 48 бітів кожна. Це так звані *циклові ключі*. Блок M розбивається на дві рівні частини по 32 біти: ліву L_0 та праву R_0. Відбувається 16

циклів перетворення повідомлення $M = L_0R_0$. У i-му циклі (L_{i-1}, R_{i-1}) за допомогою K_i перетворюється у (L_i, R_i) наступним чином:

$$L_i = R_{i-1},$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i),$$

де f — деяка функція, що перетворює двійкові послідовності довжини 80 у двійкові послідовності довжини 32. Насамкінець ліва та права частини міняються місцями, що і є результатом роботи алгоритму: $E_K(M) = R_{16}L_{16}$.

Неважко перевірити (вправа 3.6), що описана процедура різні блоки M' і M'' перетворює у різні блоки $E_K(M')$ і $E_K(M'')$, незалежно від вибору функції f. Більше того, якщо $E_{K_1,\dots,K_{16}}(M) = C$, то $M = E_{K_{16},\dots,K_1}(C)$, тобто дешифруючий алгоритм ідентичний із шифруючим, варто лише взяти послідовність ключів у зворотньому порядку.

Функцію f стандарт DES пропонує такої форми: $f(R, K_i) = s(e(R) \oplus K_i)$. Функція e розширює R із 32 бітів до 48, вставляючи копії деяких 16 позицій. Функція s перетворює двійкові послідовності довжини 48 у двійкові послідовності довжини 32. Вона реалізується так. Двійковий вхід A довжини 48 розбивається на 8 блоків $A_1 \dots A_8$ по 6 бітів і кожен блок A_i замінюється на деякий блок із 4 бітів шляхом застосування деякої функції s_i. Кожна із цих 8-ми функцій реалізована незалежно від інших. Реалізації цих функцій називаються *S-блоками.* Після цього отримані 32 біти перемішуються згідно із деякою перестановкою p. Таким чином, $s(A) = p(s_1(A_1) \dots s_8(A_8))$, що й завершує наш опис алгоритму DES.

Деякі моменти нами були опущені. Згідно із стандартом, описана нами процедура починається із переставлення бітів блоку M відповідно до фіксованої перестановки IP і завершується застосуванням до отриманого результату оберненої перестановки IP^{-1}.

Із 64 бітів ключа K алгоритм DES насправді використовує лише 56. Решта бітів можуть служити для перевірки неушкодженості даних при пересиланні ключа — з цією метою кожен восьмий біт покладають рівним сумі за модулем 2 попередніх семи. Кожен із циклових ключів $K_1, \dots, K_{16}$ отримується проектуванням K на деякі 48 координат, які окрім того переставляються місцями.

Кількість циклів у алгоритмі DES дорівнює 16. Чим це краще від, скажімо, двох циклів? Справа в тому, що у випадку двох циклів незначна зміна ключа (наприклад, у кількох бітах) спричиняє таку ж незначну зміну у криптотексті. Ця обставина звужує множину ключів, які досить перебрати для реконструкції початкового тексту. Кількість циклів збільшується для того, щоб кожен біт криптотексту залежав від всіх бітів ключа та від всіх бітів відкритого тексту.

3.5. Рецепти використання блокових шифрів. Шифрування блоками по 64, про яке йшлося у попередньому пункті, називається *режимом простого заміщення* або *режимом електронної кодової книжки.* Певні недоліки такого способу шифрування лежать на поверхні. Зрозуміло, що однакові блоки у відкритому тексті перейдуть у однакові блоки криптотексту — отже деяка інформація про структуру повідомлення все ж буде доступною для суперника. Додатковим внеском у криптоаналіз може бути те, що суперник може знати кілька перших чи останніх блоків відкритого тексту (наприклад, "Вельмишановний Іване Івановичу" чи "Щиро Ваш, Петро Петрович").

Нема гарантій від неприємностей ще й такого гатунку. Суперник може перехопити повідомлення і частково його сфальшувати шляхом заміни деяких блоків іншими. Наприклад, у банківському розпорядженні перерахувати значні кошти на певний рахунок зловмисник може змінити номер рахунку на власний. Для цього йому слід раніше переказати самому собі цілком легально невелику суму грошей, і перехопивши відповідний фінансовий документ, дізнатися з нього номер свого рахунку, зашифрований ключем, що використовується банком того дня.

Всіх цих небезпек можна уникнути, здійснюючи шифрування в інших режимах. Ці режими передбачені стандартом для алгоритму DES, але вони однаково придатні й для інших блокових шифрів.

Зчеплення зашифрованих блоків. До кожного блоку M_i, перед подачею його на вхід алгоритму E, додається за модулем 2 зашифрований попередній блок C_{i-1}. Таким чином,

$$C_i = E_K(M_i \oplus C_{i-1}).$$

Ясно, що тепер однаковим блокам у відкритому тексті будуть відповідати різні блоки криптотексту. Кожен блок криптотексту у цьому режимі залежить від всіх попередніх, тому підміна якогось блоку повідомлення неминуче буде виявлена адресатом.

Все ж, якщо два різні повідомлення починаються однаковим блоком, відповідні криптотексти теж будуть мати перший блок однаковий. У певних ситуаціях, це може дати небажану інформацію суперникові. Щоб позбутися цього недоліку, перед шифруванням до повідомлення приписують якийсь випадковий блок — так званий *ініційний вектор.*

Зворотний зв'язок за виходом. У цьому режимі блоковий алгоритм E служить для породження блоків *псевдовипадкових бітів* B_i довжини 64: $B_i = E_K(I_i)$, де I_1 — довільний ініційний вектор, а I_i отримується з I_{i-1} відкиданням перших k бітів і приписуванням справа перших k бітів блоку B_{i-1}. Тут k є фіксованим натуральним числом, меншим від 64. Блок повідомлення M_i передається у вигляді криптотексту $C_i = M_i \oplus B_i$.

Зворотний зв'язок за криптотекстом. Цей режим схожий з попереднім, лише для поновлення I_i використовується C_{i-1} замість B_{i-1}.

ВПРАВИ

3.1. a) Подати у двійковій формі слова ананас та яблуко.

b) Зашифрувати повідомлення ананас, використавши шифр одноразового блокноту з ключем 110000 011110 010100 110010 010110 011110.

c) З яким ключем той же результат дало б шифрування повідомлення яблуко?

3.2. Із скількома різними ключами можна зашифрувати двійкове повідомлення довжини n шифром одноразового блокноту?

3.3. a) Зашифрувати за допомогою шифру ADFGVX з ключем, зображеним на малюнку 7, повідомлення you must strike while the iron is hot.

b) Дешифрувати криптотекст avxdx gddfa xavxx fxfvx ffgxf fxgga aaxaa agfgx gg, отриманий з допомогою того ж шифру з тим же ключем.

3.4. Нехай (E, D) — алгоритми шифрування і дешифрування деякої криптосистеми. Криптотекст C отримано з повідомлення M за допомогою ключів K_1, K_2 і K_3 так: $C = E_{K_3}(D_{K_2}(E_{K_1}(M)))$. Яким чином можна зробити зворотнє перетворення, тобто за C отримати M?

3.5. Чи утворюють групу шифри:

a) частоколу;

b) Скитала;

c) матричний шифр обходу;

d) Кардано?

3.6. Позначимо через E^i алгоритм DES з i циклами, описаний на сторінці 42 для 16 циклів. Довести, що для довільної функції f, якщо $E^i_{K_1,\dots,K_i}(M) = C$, то $M = E^i_{K_i,\dots,K_1}(C)$

a) для $i = 2$;

b) для довільного значення i.

3.7. ([136]) Для двійкового слова X через $\overline{X}$ будемо позначати результат заміни кожного біта у слові на протилежний. Наприклад, $\overline{101} = 010$. Нехай E — алгоритм шифрування стандарту DES. Довести, що $E_{\overline{K}}(\overline{M}) = \overline{E_K(M)}$ для довільних повідомлення M та ключа K.

3.8. Скількома способами можна було б вибрати функцію s, що фігурує у нашому описі алгоритму DES? (Стандарт задає її явним чином.)

3.9. a) Довести, що якби S-блоки s_i, де $i \le 8$, були вибрані лінійними функціями з $(\mathbb{Z}_2)^6$ в $(\mathbb{Z}_2)^4$, то DES утворював би групу, яка була б підгрупою групи невироджених лінійних операторів над $(\mathbb{Z}_2)^{64}$.

b) Знайти порядок згаданої групи невироджених лінійних операторів над $(\mathbb{Z}_2)^{64}$.

3.10. Вказати порядок дешифрування повідомлення із m блоків для різних режимів застосування блокових шифрів.

ЛІТЕРАТУРА

Вагомим внеском у криптографію періоду, розглянутого в цьому параграфі, є криптографічні дослідження Клода Шеннона [60]. У цій роботі, зокрема, розглядаються поняття композиції шифрів та надійності в теоретико-інформаційному сенсі.

В [89, 116] доведено, що композиція двох шифрів не може бути криптографічно слабшою від кожного з них зокрема, за умови, що ключі вибираються випадково і незалежно один від одного.

Повний опис стандарту DES і його аналіз можна знайти в [121] (див. також [137, 53, 56]). 56-бітова довжина ключа зазнала критики з самого початку впровадження стандарту. У [86] підраховано, що коштом \$20 млн. можна створити машину із 2^{20} паралельних процесорів, кожен з яких у стані випробувати 2^{20} ключів за секунду. Очевидно, така машина розкривала б шифр щонайбільше за 2^{16} секунд, тобто менш як за добу. Див. також [104] та обговорення в [43]. Зрозуміло, що перебірні можливості сучасної техніки є ще більшими. Тому, коли є підстави остерігатися дуже серйозного криптоаналізу, DES застосовують двічі або тричі із незалежним вибором ключів. Факт, що DES не утворює групи, встановлений в [82]. Атаки, ефективніші за брутальну, запропоновані в [75] (з вибором відкритого тексту) та [118] (за відомим відкритим текстом).

Режими застосування блокових шифрів висвітлені в [57].

Розділ II.

Елементарна криптографія II: математичний підхід

Звід математичних понять та теорем, що зустрічаються в цьому розділі, міститься в додатку Б.

§ 1. Формалізм

1.1. Абетка II. *Алфавітом* називаємо довільну скінченну множину. Типовими прикладами, які зустрічаються у нашому курсі, є

$\{\text{а},\text{б},\text{в},\text{г},\ldots,\text{ь},\text{ю},\text{я}\}$ — український алфавіт,
$\{\sqcup,\text{а},\text{б},\text{г},\ldots,\text{ь},\text{ю},\text{я}\}$ — він же з пропуском між словами,
$\{\sqcup,\cdot,,,?,\ldots,\text{ь},\text{ю},\text{я}\}$ — він же з розділовими знаками,
$\{0,1\}^6$ — всі 0-1-послідовності довжини 6,
$\mathbb{Z}_{33}=\{0,1,2,\ldots,32\}$ — кільце лишків за модулем 33.

Алфавіти позначатимемо рукописними літерами $\mathcal{A}$, $\mathcal{B}$ і т.д. Елементи алфавіту називатимемо *символами* або *буквами*, навіть якщо це буде розходитись із звичним вживанням останнього терміну. Наприклад, і цифра 1, і число 10 є буквами алфавіту $\mathbb{Z}_{33}$. *Слово в алфавіті* $\mathcal{A}$ — це скінченна послідовність букв цього алфавіту. $\mathcal{A}^*$ служитиме позначенням для множини всіх слів у алфавіті $\mathcal{A}$. *Довжина слова* — це кількість букв у ньому. Зрозуміло, що множина слів у алфавіті $\mathcal{A}$ довжини l є нічим іншим як декартовим степенем $\mathcal{A}^l$. Довжину слова w позначатимемо як $|w|$. Для слів v і u через vu позначатимемо результат їх злиття

в одне слово, саме в такому порядку. Слово v називатимемо *префіксом* слова w у випадку, коли $v=w$ або $w=vu$ для деякого слова u.

Коли говорять про *криптосистему* чи *шифр*, мають на увазі такі об'єкти:

- Алфавіт $\mathcal{A}$, в якому записуються *повідомлення (відкриті тексти)*. Повідомлення M є словом в алфавіті $\mathcal{A}$ (яке може складатися з багатьох слів у звичному лінгвістичному розумінні), тобто $M\in\mathcal{A}^*$. Множина $\mathcal{A}^*$ називається *простором повідомлень* або *відкритих текстів*.
- Алфавіт $\mathcal{B}$, в якому записуються *криптотексти*. Множина $\mathcal{B}^*$ називається *простором криптотекстів*. Часто $\mathcal{A}=\mathcal{B}$.
- Простір ключів $\mathcal{K}$, який складається із слів у деякому алфавіті, що називаються *ключами*.
- *Шифруюче відображення* $E:\mathcal{K}\times\mathcal{A}^*\to\mathcal{B}^*$.
- *Дешифруюче відображення* $D:\mathcal{K}\times\mathcal{B}^*\to\mathcal{A}^*$. Відображення E і D повинні мати таку властивість:

$$D(K,E(K,M))=M \text{ для всіх } M\in\mathcal{A}^* \text{ і } K\in\mathcal{K}. \tag{1}$$

Фіксація ключа K визначає відображення $E_K:\mathcal{A}^*\to\mathcal{B}^*$ за формулою $E_K(M)=E(K,M)$ для кожного $M\in\mathcal{A}^*$. Подібним чином задається й відображення $D_K:\mathcal{B}^*\to\mathcal{A}^*$.

Властивість (1) по-іншому можна сформулювати так: для будь-якого K дешифруюче відображення $D_K:\mathcal{B}^*\to\mathcal{A}^*$ є оберненим зліва до шифруючого відображення $E_K:\mathcal{A}^*\to\mathcal{B}^*$. З теореми про обернене відображення (див. додаток Б) випливає, що E_K мусить бути ін'єкцією. Цілком зрозуміло, що ця умова рівнозначна принциповій можливості дешифрування. Навпаки, на підставі тієї ж теореми будь-яка родина ін'єктивних відображень $\{E_K:\mathcal{A}^*\to\mathcal{B}^*\}_{K\in\mathcal{K}}$ може розглядатись як шифруюча в тому плані, що для них існують обернені зліва відображення $\{D_K:\mathcal{B}^*\to\mathcal{A}^*\}_{K\in\mathcal{K}}$, які можуть вважатися дешифруючими.

Дуже часто у конкретних криптоситемах шифруюче відображення E_K крім ін'єктивності володіє також сюр'єктивністю. За теоремою про обернене відображення, дешифруюче відображення D_K є оберненим до E_K не тільки зліва, а й справа — обставина, що має криптографічні застосування. Таким чином, у цьому випадку E_K і D_K є взаємно оберненими:

$$D_K(E_K(M))=E_K(D_K(M))=M.$$

Криптосистему називаємо *ефективною*, якщо шифруюче і дешифруюче відображення реалізуються швидкими алгоритмами. Навряд

чи для читача хоча б з невеликим досвідом програмування таке формулювання потребує уточнення. Ті ж читачі, яких турбує дещо неформальний характер поняття ефективності, повинні отримати певне полегшення від ознайомлення з розділом III.

Питання, які криптосистеми слід називати *надійними,* детально обговорювалось у попередньому розділі і залишатиметься предметом прискіпливого розгляду й надалі.

Шифр називається *блоковим* з періодом l, якщо шифруюче відображення задається спочатку на словах довжини l, тобто маємо $E : \mathcal{K} \times \mathcal{A}^l \to \mathcal{B}^*$, а після цього поширюється на слова довільної довжини наступним чином. Якщо $M = M_1 M_2 \dots M_t$, де *блоки* M_i, $i \le t$, мають довжину l, то $E(K, M) = E(K, M_1)E(K, M_2)\dots E(K, M_t)$. Якщо ж останній блок M_t має меншу від l довжину, то він доповнюється до повного довільним наперед обумовленим чином.

Часто у блоковому шифрі шифруюче відображення зберігає довжину, тобто є вигляду $E : \mathcal{K} \times \mathcal{A}^l \to \mathcal{B}^l$. Окрім того множини $\mathcal{A}^l$ і $\mathcal{B}^l$ можуть містити однакову кількість слів, як це буде, скажімо, в поширеному випадку $\mathcal{A} = \mathcal{B}$. Якщо ми маємо відображення саме такого виду і розглядаємо питання, чи можна його використовувати як шифруюче, то в пригоді знову стає теорема про обернене відображення. Згідно з нею досить довести якусь одну з чотирьох умов: 1) ін'єктивність, 2) сюр'єктивність, 3) існування оберненого зліва відображення, 4) існування оберненого справа відображення.

1.2. Дешифрування ітераціями. В цьому пункті ми розглядаємо блоковий шифр із шифруючим відображенням виду $E : \mathcal{K} \times \mathcal{A}^l \to \mathcal{A}^l$ (алфавіт криптотекстів збігається з алфавітом повідомлень). Зрозуміло, що для кожного ключа K шифруюче відображення E_K є елементом групи $\operatorname{Sym}\mathcal{A}^l$, де $\operatorname{Sym} X$ позначає групу перестановок множини X. Принагідно означимо формально поняття, введене на стор. 40: шифр $E : \mathcal{K} \times \mathcal{A}^l \to \mathcal{A}^l$ *утворює групу,* якщо родина $\{E_K\}_{K \in \mathcal{K}}$ є в $\operatorname{Sym}\mathcal{A}^l$ підгрупою.

Шифруюче відображення $E_K : \mathcal{A}^l \to \mathcal{A}^l$ можна застосовувати кілька разів. Відображення $E_K^s : \mathcal{A}^l \to \mathcal{A}^l$ означимо індуктивно: $E_K^1 = E_K$ і $E_K^i = E_K \circ E_K^{i-1}$ для $i > 1$. E_K^i будемо називати *i-кратною ітерацією* або *i-тим степенем* відображення E_K.

Методом ітерацій суперник може скористатися у разі, коли він має доступ до шифруючого відображення із фіксованим ключем (хоча не знає, який саме ключ K "зашито" в алгоритм). Підслухавши криптотекст $C = E_K(M)$, суперник може спробувати знайти повідомлення M, обчислюючи послідовність $E_K^1(C), E_K^2(C), E_K^3(C), \dots$ доти, поки на якомусь m-му кроці не виявиться, що $E_K^m(C) = C$. Це означає, що повідомлення було отримане на попередньому кроці: $M = E_K^{m-1}(C)$.

Твердження 1.1. *Для шифруючого відображення $E : \mathcal{K} \times \mathcal{A}^l \to \mathcal{A}^l$ метод ітерацій через деяку кількість кроків приводить до успіху. Точніше, якщо алфавіт складається з n букв, то для будь-якого ключа K існує число $m \le (n^l)!$ таке, що $E_K^m(C) = C$ для всіх $C \in \mathcal{A}^l$.*

Доведення. В якості m можна взяти порядок відображення E_K як елемента групи $\operatorname{Sym}\mathcal{A}^l$, який за теоремою Лагранжа не перевищує $(n^l)!$, порядку цієї групи (див. додаток Б). Тоді для будь-якого $C \in \mathcal{A}^l$ матимемо $E_K^m(C) = \mathrm{id}_{\mathcal{A}^l}(C) = C$. ∎

ВПРАВИ

1.1. Дати формальне означення композиції двох шифруючих відображень $E_1 : \mathcal{K}_1 \times \mathcal{A}^* \to \mathcal{B}^*$ і $E_2 : \mathcal{K}_2 \times \mathcal{B}^* \to \mathcal{C}^*$ (повідомлення шифрується спочатку за допомогою E_1, а потім ще раз за допомогою E_2).

1.2. Дати незалежне від теореми Лагранжа доведення того, що метод ітерацій раніше чи пізніше досягає успіху.

1.3. Довести підсилення твердження 1.1. А саме, для будь-якого ключа метод ітерацій приведе до успіху за не більш як

а) $e^{\frac{n^l}{e}}$ кроків, де $e = 2,71828\dots$ — основа натуральних логарифмів.

b) $n^{l\pi(n^l)}$ кроків, де $\pi(x)$ — кількість простих чисел, що не перевищують x (див. пункт IV.1.3).

1.4. Довести, що для розкриття шифру Віженера над n-символьним алфавітом досить n ітерацій.

§ 2. Арифметика

2.1. Алгоритм Евкліда. Для будь-якого цілого a і натурального b однозначно визначені цілі числа q і r такі, що $a = bq + r$ і $0 \le r < b$. Число q називається *часткою,* а r — *остачею* від ділення a на b. Наприклад, рівність $-20 = (-1) \cdot 67 + 47$ означає, що -20 при діленні на 67 дає частку -1 і остачу 47. Для остачі будемо вживати таке позначення: $r = a \bmod b$. Число r будемо також називати *(зведеним) лишком* числа a за модулем b. Якщо $r = 0$, то кажуть, що a *ділиться на* b (*націло* або *без остачі*) і пишуть $b \mid a$. Кажуть також, що b *ділить* a, і називають b *дільником* числа a, а a — *кратним* числа b. Запис $b \nmid a$ означатиме,

що a не ділиться на b. Для цілих a і b через НСД(a,b) позначається їх *найбільший спільний дільник (НСД)* — найбільше з-поміж таких c, що одночасно $c \mid a$ і $c \mid b$ (хоча б одне із a і b вважається відмінним від нуля). Числа a і b називаються *взаємно простими*, якщо НСД$(a,b)=1$.

Алгоритм Евкліда (3 ст. до н.е.) для знаходження НСД двох натуральних чисел a і b грунтується на співвідношеннях

$$\text{НСД}(a,b) = \text{НСД}(a, a \bmod b) \text{ для } a \geq b \tag{1}$$

$$\text{НСД}(a,0) = a \tag{2}$$

Спершу продемонструємо ідею цього алгоритму на прикладі.

Приклад 2.1. Щоб знайти НСД$(211,89)$, застосовуємо послідовно (1) аж доки не опинимось у ситуації, коли можна використати (2). Таким чином робота алгоритму зводиться до кількаразового ділення з остачею.

$$\begin{aligned} 211 &= 79\cdot 2+53\\ 79 &= 53\cdot 1+26\\ 53 &= 26\cdot 2+1\\ 26 &= 1\cdot 26+0 \end{aligned}$$

Маємо НСД$(211,79)$ = НСД$(79,53)$ = НСД$(53,26)$ = НСД$(26,1)$ = НСД$(1,0)=1$.

Опишемо тепер, що відбувається в загальному випадку.

Алгоритм Евкліда обчислення НСД(a,b), $a \geq b$.

- Покласти $r_0 = a$, $r_1 = b$, $i = 1$.
- Виконати i-ий крок, що полягає у діленні з остачею:

$$r_{i-1} = r_i \cdot q_i + r_{i+1}. \tag{3}$$

- Якщо $r_{i+1} > 0$, то збільшити i на 1 і перейти до виконання наступного кроку згідно з попереднім пунктом. Якщо $r_{i+1} = 0$, то завершити роботу із результатом НСД$(a,b) = r_i$.

Коректність. Алгоритм завершує роботу як тільки $r_{i+1}=0$. Рано чи пізно це трапиться, оскільки $0 \leq r_{i+1} < r_i$ для кожного i. Припустимо так сталося на m-ому кроці, тобто $r_{m+1}=0$. Результатом роботи алгоритму є значення r_m. Рівність $r_m = \text{НСД}(a,b)$ випливає з таких міркувань. Співвідношення НСД$(a,b) = \text{НСД}(r_{i-1}, r_i)$ для $1 \leq i \leq m+1$ доводиться індукцією з використанням (1). При $i = m+1$ із врахуванням (2) отримуємо НСД$(a,b) = \text{НСД}(r_m, 0) = r_m$.

Ефективність. Під ефективністю алгоритму Евкліда розуміємо те, що кількість кроків m, які виконуються на вході a,b, не є надто великою. Із співвідношення $0 \leq r_{i+1} < r_i$ випливає, що $m \leq b$. Однак

ця оцінка є незадовільною. Скажімо, для чисел a і b з якоюсь сотнею цифр у десятковому записі кожного, нам гарантовано лише, що на знаходження їх НСД буде витрачено не більше 10^{100} кроків. Але ж найбільш швикодіючій ЕОМ для виконання такої кількості кроків потрібен фантастично великий обсяг часу.

На щастя, оцінку на m можна суттєво поліпшити. Покажемо, що $m \leq 2\log_2 b + 1$. Для цього використаємо нерівність $r_{i+1} < r_{i-1}/2$. Вона доводиться розглядом двох випадків: якщо $r_i < r_{i-1}/2$, то використовуємо нерівність $r_{i+1} < r_i$; якщо ж $r_i \geq r_{i-1}/2$, то використовуємо рівність $r_{i+1} = r_{i-1} \bmod r_i$. Таким чином, кожні два кроки зменшують r_{i+1} принаймні вдвічі, і не пізніше, ніж за $2\log_2 b+1$ кроків ми прийдемо до $r_{i+1}=0$.

Алгоритм Евкліда дає такий наслідок.

Твердження 2.2. *Для кожної пари взаємно простих чисел a і b можна знайти такі цілі u і v, що $ua+vb=1$.*

Доведення. За умовою НСД$(a,b)=1$. Тому на передостанньому кроці алгоритму Евкліда матимемо $r_{m-2} = r_{m-1}q_{m-1}+1$. Звідси $1 = 1\cdot r_{m-2} + (-q_{m-1})r_{m-1}$. Використавши це як базу для зворотньої індукції за i від $m-1$ до 1, доведемо, що $1 = u_i r_{i-1} + v_i r_i$ для деяких цілих u_i та v_i. Справді, якщо $1 = u_{i+1}r_i + v_{i+1}r_{i+1}$, то після підстановки замість r_{i+1} його значення, визначеного з (3), маємо $1 = v_{i+1}r_{i-1} + (u_{i+1} - q_i v_{i+1})r_i$.

При $i=1$ отримуємо потрібне твердження. ∎

Зазначимо, що алгоритм Евкліда дає ефективний спосіб знаходження коефіцієнтів u і v для заданої пари a,b.

Приклад 2.3. Нехай $a=211$, $b=79$. Протокол роботи алгоритму Евкліда виписаний у прикладі 2.1. Рухаючись знизу вверх, отримуємо

$$\begin{aligned} 1 &= 1\cdot 53 + (-2)\cdot 26 = 1\cdot 53 + (-2)\cdot(79 - 1\cdot 53) = (-2)\cdot 79\\ &\quad +3\cdot 53 = (-2)\cdot 79 + 3\cdot(211 - 2\cdot 79) = 3\cdot 211 + (-8)\cdot 79. \end{aligned}$$

Отже, $u=3$ і $v=-8$. Продемонстрований спосіб визначення коефіцієнтів u і v є наочним, але не оптимальним, бо вимагає збереження в пам'яті проміжних обчислень алгоритму Евкліда. Кращий спосіб, так званий *розширений алгоритм Евкліда,* пропонується у вправі 2.7.

2.2. Розклад на прості співмножники. Очевидно, що кожне натуральне число a має принаймні два дільники, а саме 1 і a. Ці дільники називаються *тривіальними.* Якщо число a не має ніяких інших дільників, то воно називається *простим,* в іншому ж разі — *складеним.*

Твердження 2.4. *Якщо просте число p ділить добуток ab двох натуральних чисел, то воно мусить ділити хоча б одне із чисел a і b.*

Доведення. Якщо a не ділиться на p, то a і p взаємно прості. Тоді за твердженням 2.2 для деяких цілих чисел u і v виконується рівність $ua + vp = 1$. Домноживши її на b, отримаємо $uab + vbp = b$. Оскільки $p \mid ab$, то ліва частина ділиться на p, а відтак $p \mid b$. ∎

Зрозуміло, що кожне складене число можна записати як добуток простих. Такий добуток називається *розкладом числа на прості співмножники*. Вважаємо, що розклад простого числа складається з єдиного елемента — самого числа. Має місце

Теорема про однозначність розкладу на прості співмножники. *Кожне більше від 1 ціле число однозначно розкладається на прості співмножники (якщо не враховувати їхнього порядку).*

Доведення. Припустимо, що $\prod_{i=1}^{k} p_i^{\alpha_i} = \prod_{i=1}^{k} p_i^{\beta_i}$ — два різні розклади одного й того ж числа на прості співмножники. Тоді мусить бути $\alpha_j \neq \beta_j$ для деякого j. Якщо $\alpha_j > \beta_j$, то маємо $p_j \mid \prod_{i \neq j} p_i^{\beta_i}$, що суперечить твердженню 2.4. Випадок $\alpha_j < \beta_j$ розглядається симетрично. ∎

Розклад, в якому прості співмножники йдуть у неспадному порядку, називається *канонічним*.

2.3. Конгруенції. В пункті 2.1 ми домовились позначати остачу від ділення цілого a на натуральне b через $a \bmod b$. Позначення mod ми будемо вживати і в іншому значенні. А саме, ми будемо писати $x \equiv y \pmod{n}$, якщо цілі x і y при діленні на натуральне n дають однакову остачу (тобто, $x \bmod n = y \bmod n$). Такі x і y називатимемо *конгруентними* або *рівними за модулем n*. Відношення між x і y називається *конгруенцією* або *порівнянням*.

Твердження 2.5. *Наступні три умови еквівалентні.*

1) $x \equiv y \pmod{n}$.
2) $x = y + kn$ *для деякого цілого* k.
3) $n \mid (x - y)$. ∎

Твердження 2.6. *Конгруенції мають такі властивості.*

1) *Це відношення еквівалентності:*
   a) $x \equiv x \pmod{n}$ *(рефлексивність);*
   b) якщо $x \equiv y \pmod{n}$, *то* $y \equiv x \pmod{n}$ *(симетричність);*
   c) якщо $x \equiv y \pmod{n}$ *і* $y \equiv z \pmod{n}$, *то* $x \equiv z \pmod{n}$ *(транзитивність).*

2) *Конгруенції можна почленно додавати: якщо* $x_1 \equiv y_1 \pmod{n}$ *і* $x_2 \equiv y_2 \pmod{n}$, *то* $x_1 + x_2 \equiv y_1 + y_2 \pmod{n}$. *Зокрема, до обох частин конгруенції можна додавати одне й те ж число.*
3) *Конгруенції можна почленно перемножувати: якщо* $x_1 \equiv y_1 \pmod{n}$ *і* $x_2 \equiv y_2 \pmod{n}$, *то* $x_1 x_2 \equiv y_1 y_2 \pmod{n}$. *Зокрема, обидві частини конгруенції можна домножувати на одне й те ж число.*
4) *Обидві частини конгруенції можна скорочувати на їхній спільний дільник, якщо він взаємно простий з модулем: якщо* $d \mid x$, $d \mid y$ *і* НСД$(d, n) = 1$, *то з* $x \equiv y \pmod{n}$ *випливає* $x/d \equiv y/d \pmod{n}$.
5) *Обидві частини конгруенції і модуль можна скорочувати на їхній спільний дільник: якщо* $d \mid x$, $d \mid y$ *і* $d \mid n$, *то з* $x \equiv y \pmod{n}$ *випливає* $x/d \equiv y/d \pmod{n/d}$.
6) *Якщо* $m \mid n$, *то з* $x \equiv y \pmod{n}$ *випливає* $x \equiv y \pmod{m}$.
7) *Для p і q простих,* $x \equiv y \pmod{pq}$ *тоді і лише тоді, коли одночасно* $x \equiv y \pmod{p}$ *і* $x \equiv y \pmod{q}$. ∎

Приклад 2.7. Уникаючи обчислювальної роботи, покажемо, що 73524 ділиться на 11. Очевидно, $10 \equiv -1 \pmod{11}$. Множачи цю конгруенцію саму на себе, отримаємо: $100 \equiv 1 \pmod{11}$, $1000 \equiv -1 \pmod{11}$, $10000 \equiv 1 \pmod{11}$. Після домноження на числа 2, 5, 3, 7 відповідно, матимемо: $20 \equiv -2 \pmod{11}$, $500 \equiv 5 \pmod{11}$, $3000 \equiv -3 \pmod{11}$, $70000 \equiv 7 \pmod{11}$. Почленне сумування дасть $73520 \equiv 7 \pmod{11}$. Додавши до обох частин 4, отримаємо $73524 \equiv 11 \equiv 0 \pmod{11}$.

2.4. Кільце лишків. Для натурального n через $\mathbb{Z}_n$ позначаємо множину $\{0, 1, \ldots, n-1\}$, наділену операціями додавання та множення за модулем n. Сумою x і y із $\mathbb{Z}_n$ є $(x+y) \bmod n$, а їх добутком є $(x \cdot y) \bmod n$. Відносно цих операцій $\mathbb{Z}_n$ є комутативним кільцем з одиницею, яке називається *кільцем зведених лишків за модулем n*. Через $\mathbb{Z}_n^*$ позначаємо мультиплікативну групу елементів, для яких в $\mathbb{Z}_n$ є обернені відносно множення.

Твердження 2.8. $\mathbb{Z}_n^*$ *складається з елементів x, взаємно простих з n, і лише з них.*

Доведення. Якщо НСД$(x, n) = 1$, то за твердженням 2.2 маємо $ux + vn = 1$ для деяких цілих u і v. Звідси $ux = 1 \pmod{n}$ і $x' = u \bmod n$ є оберненим за множенням до x в $\mathbb{Z}_n$.

Навпаки, якщо $xx' = 1$ в $\mathbb{Z}_n$, то добуток x та x' як цілих чисел дає остачу 1 при діленні на n: $xx' = qn + 1$. Отже, кожен спільний дільник чисел x та n ділить також 1, звідки НСД$(x, n) = 1$. ∎

З доведення твердження бачимо, що розширений алгоритм Евкліда дає ефективний спосіб знаходження оберненого елемента для будь-якого заданого $x \in \mathbb{Z}_n^*$.

Елемент, обернений до $x \in \mathbb{Z}_n^*$ відносно множення, будемо позначати через $x^{-1} \bmod n$ або просто x^{-1}. Ділення на x в $\mathbb{Z}_n$ означатиме множення на x^{-1}. Елемент, обернений до $x \in \mathbb{Z}_n$ відносно додавання, будемо позначати через $-x$. Зокрема, $-1 = n-1$ в $\mathbb{Z}_n$. Як звичайно, в $\mathbb{Z}_n$ можна ввести операцію віднімання: $x - y = x + (-y)$.

Приклад 2.9. Нехай ми хочемо знайти елемент, обернений до 79 в $\mathbb{Z}_{211}^*$. У прикладі 2.3 була отримана рівність $1 = 3 \cdot 211 + (-8) \cdot 79$. З неї негайно випливає $(-8) \cdot 79 \equiv 1 \,(\mathrm{mod}\, 211)$. Отже, $79^{-1} \bmod 211 = (-8) \bmod 211 = 203$.

Наслідок 2.10. *Для простого модуля p, кільце $\mathbb{Z}_p$ є полем.* ∎

Через $\phi(n)$ позначаємо порядок групи $\mathbb{Z}_n^*$. Іншими словами, значення $\phi(n)$ дорівнює кількості натуральних чисел, що не перевищують n і взаємно прості з n. ϕ називається *функцією Ойлера.*

Теорема Ойлера (1763). *Для взаємно простих цілого x і натурального n справедлива конгруенція $x^{\phi(n)} \equiv 1 \,(\mathrm{mod}\, n)$.*

Доведення. Припустимо $1 \le x < n$ і розглянемо x як елемент мультиплікативної групи $\mathbb{Z}_n^*$. За теоремою Лагранжа порядок елемента x є дільником порядку групи, $\phi(n)$ в нашому випадку. Тому $x^{\phi(n)} = 1$ в $\mathbb{Z}_n^*$, звідси й випливає теорема.

Випадок довільного x зводимо до попереднього, використавши конгруенцію $x^{\phi(n)} \equiv (x \bmod n)^{\phi(n)} \,(\mathrm{mod}\, n)$. ∎

Як легко бачити, $\phi(p) = p - 1$ для простого p. Звідси отримуємо такий наслідок теореми Ойлера.

Мала теорема Ферма (1640). *Якщо ціле x не ділиться на просте p, то $x^{p-1} \equiv 1 \,(\mathrm{mod}\, p)$.* ∎

Китайська теорема про остачі (I ст. до н.е.). *Для будь-якої пари взаємно простих натуральних чисел n_1 і n_2 та для будь-якої пари цілих чисел x_1 і x_2, можна знайти таке ціле x, де $x \equiv x_1 \,(\mathrm{mod}\, n_1)$ і $x \equiv x_2 \,(\mathrm{mod}\, n_2)$.*

Доведення. За твердженням 2.2 для деяких цілих u_1 і u_2 маємо рівність $u_1 n_1 + u_2 n_2 = 1$, з якої випливають співвідношення $u_1 n_1 \equiv 1 \,(\mathrm{mod} n_2)$ і $u_2 n_2 \equiv 1 \,(\mathrm{mod} n_1)$. Використовуючи останні, легко пересвідчитись, що $x = x_2 u_1 n_1 + x_1 u_2 n_2$ задовільняє потрібні умови. ∎

Як видно з доведення, x для заданих n_1, n_2, x_1 і x_2 знаходиться ефективно за допомогою розширеного алгоритму Евкліда.

Приклад 2.11. Нехай ми хочемо знайти ціле x, яке при діленні на 211 давало б остачу 100, а при діленні на 79 остачу 10. У прикладі 2.3 була отримана рівність $1 = 3 \cdot 211 + (-8) \cdot 79$. Отже, в якості x можна взяти $10 \cdot 3 \cdot 211 + 100 \cdot (-8) \cdot 79 = -56870$. Зрозуміло, що це число можна замінити його остачею від ділення на $211 \cdot 79 = 16669$. В результаті отримуємо 9806.

Твердження 2.12. *Нехай $n = n_1 n_2$, де n_1 і n_2 взаємно прості. Тоді відображення $f : \mathbb{Z}_n \to \mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2}$, задане співвідношенням*

$$f(x) = (x \bmod n_1, x \bmod n_2), \tag{1}$$

є ізоморфізмом кілець.

Доведення. Перевірка того, що f є гомоморфізмом, зводиться до перевірки того, що відображення $f_i(x) = x \bmod n_i$ є гомоморфізмом із $\mathbb{Z}_n$ в $\mathbb{Z}_{n_i}$ для $i = 1, 2$. Останнє справді має місце, оскільки n_i ділить n.

Сюр'єктивність є безпосереднім наслідком Китайської теореми про остачі.

Ін'єктивність випливає із тривіальності ядра — найменшим натуральним числом, яке ділиться націло на кожне із взаємно простих чисел n_1 і n_2, є добуток $n_1 n_2 = n$. ∎

Слід зауважити, що як f, так і обернене відображення f^{-1} обчислюються ефективним чином (див. приклад 2.11).

Попереднє твердження разом із твердженням Б.2 дає

Наслідок 2.13. *Нехай $n = n_1 n_2$, де n_1 і n_2 взаємно прості. Тоді звуження відображення (1) на $\mathbb{Z}_n^*$ є ізоморфізмом із цієї групи на прямий добуток груп $\mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^*$.* ∎

Наслідок 2.14. (Мультиплікативність функції Ойлера) *Для попарно взаємно простих $n_1, n_2, \ldots, n_l$ справедлива рівність $\phi(n_1 n_2 \ldots n_l) = \phi(n_1)\phi(n_2)\ldots\phi(n_l)$.*

Доведення. Випадок $l = 2$ безпосередньо випливає з наслідку 2.13. Загальне твердження доводиться індукцією за l. ∎

Формула для функції Ойлера. *Нехай $n = p_1^{\alpha_1} \cdot \ldots \cdot p_l^{\alpha_l}$ — розклад натурального числа n на прості співмножники. Тоді $\phi(n) = n(1 - 1/p_1) \cdot \ldots \cdot (1 - 1/p_l)$.*

Доведення. Спочатку розглянемо випадок $l = 1$, тобто $n = p^\alpha$ для деякого простого p. Числами, які не перевищують число p^α і не взаємно прості з ним, є $p, 2p, 3p, \ldots, p^\alpha = p^{\alpha-1}p$ — всього $p^{\alpha-1}$ штук. Звідси $\phi(p^\alpha) = p^\alpha - p^{\alpha-1} = p^\alpha(1 - 1/p)$. Для $l > 1$ формула випливає з мультиплікативності функції Ойлера. ∎

Твердження 2.15. *$\phi(n) > n/(6 \ln\ln n)$ для $n > 4$.*

Доведення. Проведене в [135]. Див. також [13, вправа II.9(g)]. ∎

2.5. Кільце матриць. Через $\mathrm{M}_k(\mathbb{Z}_n)$ позначаємо множину матриць розміру $k \times k$ з коефіцієнтами з $\mathbb{Z}_n$. З операціями додавання і множення матриць $\mathrm{M}_k(\mathbb{Z}_n)$ утворює кільце. $\mathrm{M}_k(\mathbb{Z}_n)^* = \mathrm{GL}_k(\mathbb{Z}_n)$ служить позначенням для мультиплікативної групи оборотних матриць. Порядок цієї групи позначимо через $\phi_k(n)$. Зрозуміло, що $\phi_1(n)$ є нічим іншим як функцією Ойлера. У цьому пункті ми виведемо формулу для $\phi_k(n)$, де k довільне натуральне число. Будемо діяти за тим же планом, що був реалізований для функції Ойлера.

Твердження 2.16. *Нехай $n = n_1 n_2$, де n_1 і n_2 взаємно прості. Тоді відображення $f_k : \mathrm{M}_k(\mathbb{Z}_n) \to \mathrm{M}_k(\mathbb{Z}_{n_1}) \oplus \mathrm{M}_k(\mathbb{Z}_{n_2})$, яке співставляє матриці A пару матриць A_1 та A_2, коефіцієнти яких є лишками відповідних коефіцієнтів матриці A за модулями n_1 та n_2, є ізоморфізмом кілець.*

Доведення. Очевидно, що f_k переводить одиницю кільця $\mathrm{M}_k(\mathbb{Z}_n)$ в одиницю кільця $\mathrm{M}_k(\mathbb{Z}_{n_1}) \oplus \mathrm{M}_k(\mathbb{Z}_{n_2})$. Нескладно перевірити також, що f_k зберігає операції додавання та множення; ключовим фактом при цьому є те, що n ділиться на n_1 і n_2. Отже, f_k є гомоморфізмом.

Щоб встановити сюр'єктивність, зауважимо, що f_k діє покомпонентно і на кожній із k^2 компонент це відображення є сюр'єктивним за Китайською теоремою про остачі.

Ін'єктивність випливає із тривіальності ядра. ∎

За твердженням Б.2 отримуємо

Наслідок 2.17. *Звуження відображення f_k на $\mathrm{M}_k(\mathbb{Z}_n)^*$ є ізоморфізмом із цієї групи на прямий добуток груп $\mathrm{M}_k(\mathbb{Z}_{n_1})^* \times \mathrm{M}_k(\mathbb{Z}_{n_2})^*$.* ∎

Наслідок 2.18. *Функція ϕ_k мультиплікативна: для попарно взаємно простих $n_1, n_2, \ldots, n_l$ справедлива рівність $\phi_k(n_1 n_2 \ldots n_l) = \phi_k(n_1)\phi_k(n_2)\ldots\phi_k(n_l)$.*

Доведення. Випадок $l = 2$ безпосередньо випливає з наслідку 2.17. Загальне твердження доводиться індукцією за l. ∎

Теорема 2.19. *Нехай $n = p_1^{\alpha_1} \cdot \ldots \cdot p_l^{\alpha_l}$ — розклад натурального числа n на прості співмножники. Тоді $\phi_k(n) = n^{k^2} \prod_{i=1}^{k}(1 - 1/p_1^i) \cdot \ldots \cdot \prod_{i=1}^{k}(1 - 1/p_l^i)$.*

Доведення.

Випадок 1: $n = p$ — просте.

Для $\phi_k(p)$, кількості оборотних матриць порядку k над $\mathbb{Z}_p$, нам належить показати, що

$$\phi_k(p) = p^{k^2} \prod_{i=1}^{k}(1 - 1/p^i) = \prod_{i=1}^{k}(p^k - p^{i-1}). \tag{1}$$

Будемо опиратись на той факт, що $\mathbb{Z}_p$ є полем, а відтак оборотність матриці A із $\mathrm{M}_k(\mathbb{Z}_p)$ рівносильна її невиродженості. Через $X_1, X_2, \ldots, X_k$ позначимо стовпчики матриці A. Питання, скількома способами можна вибрати матрицю в $\mathrm{M}_k(\mathbb{Z}_p)^*$, еквівалентне такому — скількома способами можна вибрати послідовність k лінійно незалежних векторів $X_1, X_2, \ldots, X_k$ в $\mathbb{Z}_p^k$?

Вектор X_1 можна вибрати довільним за винятком нульового, тобто $p^k - 1$ способом. Далі, для $1 < i \leq k$ лінійна оболонка векторів $X_1, \ldots, X_{i-1}$ є $(i-1)$-вимірним векторним підпростором простору $\mathbb{Z}_p^k$, і X_i може бути довільним вектором поза цим підпростором. $(i-1)$-вимірний підпростір містить p^{i-1} елементів, тому для вибору X_i є $p^k - p^{i-1}$ можливість. Звідси випливає (1).

Випадок 2: $n = p^\alpha$, де p просте.

Ми доведемо рівність

$$\phi_k(p^\alpha) = p^{(\alpha-1)k^2}\phi_k(p). \tag{2}$$

Легко зауважити, що з врахуванням (1) з неї випливатиме те, що потрібно.

Для $A \in \mathrm{M}_k(\mathbb{Z}_{p^\alpha})$ позначимо через $\bar{A} \in \mathrm{M}_k(\mathbb{Z}_p)$ матрицю, коефіцієнти якої є лишками за модулем p відповідних коефіцієнтів матриці A. Матриця A оборотна тоді і лише тоді, коли такою є матриця $\bar{A}$. Це випливає з тверджень Б.5 і 2.8. Справді, $\det \bar{A} = \det A \bmod p$, звідки $\mathrm{НСД}(\det A, p^\alpha) = \mathrm{НСД}(\det \bar{A}, p)$. Кількість матриць $\bar{A}$ в $\mathrm{M}_k(\mathbb{Z}_p)^*$ дорівнює $\phi_k(p)$. Кожна така матриця має $p^{(\alpha-1)k^2}$ прообразів A в $\mathrm{M}_k(\mathbb{Z}_{p^\alpha})$, які отримуються додаванням будь-якого із $p^{\alpha-1}$ чисел pj, $0 \leq j \leq p^{\alpha-1} - 1$, до кожного із k^2 коефіцієнтів матриці $\bar{A}$. Звідси маємо рівність (2).

Випадок 3: $n = p_1^{\alpha_1} \cdot \ldots \cdot p_l^{\alpha_l}$.

Цей, загальний, випадок зводиться до випадку 2 за мультиплікативністю функції $\phi_k(n)$ (наслідок 2.18). Дійсно,

$$\phi_k(n) = \prod_{j=1}^{l} \phi_k(p_j^{\alpha_j}) = \prod_{j=1}^{l} p_j^{\alpha_j k^2} \prod_{i=1}^{k}(1 - 1/p_j^i) =$$
$$\left(\prod_{j=1}^{l} p_j^{\alpha_j}\right)^{k^2} \prod_{j=1}^{l}\prod_{i=1}^{k}(1 - 1/p_j^i) = n^{k^2} \prod_{j=1}^{l}\prod_{i=1}^{k}(1 - 1/p_j^i),$$

що й слід було довести. ∎

ВПРАВИ

2.1. Обчислити а) 1212121 mod 9; b) $(-1212121) \bmod 9$.

2.2. Обчислити НСД $(959, 791)$.

2.3. Послідовність Фібоначчі задається рекурентним співвідношенням $f_1 = f_2 = 1$, $f_n = f_{n-1} + f_{n-2}$.

а) Довести, що алгоритм Евкліда при обчисленні НСД (f_n, f_{n-1}) робить $n - 2$ кроки.

b) Довести, що при обчисленні НСД (a, b), де $a \geq b$, алгоритм Евкліда робить не більше кроків, ніж при обчисленні НСД (f_n, f_{n-1}), де n таке, що $f_{n-1} \leq b < f_n$.

c) Довести, що при обчисленні НСД (a, b), де $a \geq b$, алгоритм Евкліда робить не більше, ніж $\log_\lambda b + 1$ крок, де $\lambda = (1 + \sqrt{5})/2$. Показати, що ця оцінка оптимальна з точністю до адитивної константи.

2.4. Довести, що НСД є асоціативною бінарною операцією на множині натуральних чисел.

2.5. Найменше натуральне число, яке ділиться на кожне із невід'ємних цілих чисел $a_1, a_2, \ldots, a_m$, називається їхнім *найменшим спільним кратним (НСК)* і позначається НСК $(a_1, a_2, \ldots, a_m)$. Довести, що

а) НСД $(a, b) \cdot$ НСК $(a, b) = ab$ для натуральних a і b;

b) НСК $(a, b) = ab$ для взаємно простих a і b.

2.6. Знайти цілі u і v такі, що $137u + 113v = 1$.

2.7. В позначеннях пункту 2.1 довести рівність $r_i = u_i a + v_i b$ для $0 \leq i \leq m$, де $u_0 = v_1 = 1$, $u_1 = v_0 = 0$, $u_{i+1} = u_{i-1} - q_i u_i$ і $v_{i+1} = v_{i-1} - q_i v_i$.

Позаяк $r_m =$ НСД (a, b), маємо спосіб знаходження для заданих a і b таких цілих коефіцієнтів $u = u_m$ і $v = v_m$, що НСД $(a, b) = ua + vb$. Вони обчислюються за m кроків з використанням вищенаведених рекурентних співвідношень, де q_i отримується на відповідному кроці алгоритму Евкліда. Ця процедура називається *розширеним алгоритмом Евкліда.*

2.8. Довести, що НСД (a, b) дорівнює найменшому додатному числу виду $ua + vb$, де u і v цілі.

2.9. Довести твердження 2.5 та 2.6.

2.10. За допомогою конгруенцій довести відомий критерій: число ділиться на 9 тоді і тільки тоді, коли на 9 ділиться сума цифр його десяткового запису.

2.11. Знайти канонічний розклад числа 1934064 на прості співмножники.

2.12. Знайти $8^{-1} \bmod 35$.

2.13. Знайти число, яке при діленні на 137 і 113 дає остачі 40 і 50 відповідно.

2.14. Обчислити значення $\phi(n)$ для всіх $n \leq 20$.

2.15. Для яких n значення $\phi(n)$ непарне?

2.16. Знайти всі n, для яких $\phi(n) \leq 5$.

2.17. Обчислити а) $\phi_2(26)$, b) $\phi_2(33)$.

2.18. Довести, що для довільного $y \in \mathbb{Z}_n^*$ відображення $f_y(x) = (yx) \bmod n$ є перестановкою множини $\mathbb{Z}_n$.

2.19. Матриця називається *унімодулярною*, якщо її визначник дорівнює 1 або -1 (нагадаємо, що $-1 = n - 1$ в $\mathbb{Z}_n$). Оскільки НСД $(\pm 1, n) = 1$, унімодулярні матриці є оборотними (твердження Б.5). Довести, що множина унімодулярних матриць порядку k над $\mathbb{Z}_n$ є мультиплікативною групою. Знайти порядок цієї групи.

ЛІТЕРАТУРА

Теоретико-числовий матеріал цього параграфу покривається підручниками [13, 30]. Оцінка $m \leq 2\log_2 b + 1$ на кількість кроків алгоритму Евкліда зауважена в [1]. Асимптотично кращу оцінку $m \leq 5\log_{10} b$ довів у 1844 році Ламе (див. [32, розділ 4.5.3] або [3, розділ 2.2.2], а також вправу 2.3). Низка вдосконалених варіантів алгоритму Евкліда викладена в [62, 3] та [32, розділ 4.5.2].

Чимало фактів, доведених нами для цілих чисел, поширюються на інші кільця. Найбільший спільний дільник мають будь-які два ненульові елементи в довільному *кільці головних ідеалів*. Якщо таке кільце *евклідове*, то найбільший спільний дільник знаходиться ефективно за допомогою аналогу алгоритму Евкліда. Прикладом може служити кільце многочленів від одної змінної над полем (див. пункт Б.8). Для кожного кільця головних ідеалів справедливий аналог теореми про однозначний розклад на прості множники [11, 37, 30].

У нашому викладі теорема Ойлера була виведена із теореми Лагранжа. Коротке пряме доведення наведене в [13]. Узагальнення Китайської теореми про остачі для довільних комутативних кілець з одиницею можна знайти в [37, § II.2] або [2, Тв. 12.3.1].

Обчислення функції $\phi_k(n)$ проведене в [111]. Обчислення порядків груп $\mathrm{GL}_k(F)$ і $\mathrm{SL}_k(F)$ для скінченного поля F, а також $\mathrm{GL}_k(\mathbb{Z}_{p^m})$, міститься в [26, § 11,13].

§ 3. Афінні шифри

В цьому параграфі ми розглянемо *афінні* шифри — підклас шифрів заміни, що включає як частковий випадок шифр Віженера і навіть шифр перестановки з фіксованим періодом.

3.1. Шифри простої заміни II. Зазначимо, що в рамках формалізації, впровадженої у пункті 1.1, моноалфавітні k-грамні шифри заміни можна означити як блокові шифри з періодом k. Відповідно, шифри простої заміни можна трактувати як блокові шифри з періодом 1.

Давайте повернемось ще раз до шифру зсуву, нашого першого прикладу шифру простої заміни (пункти I.1.1 та I.2.1). Подивимось, як можна описати цей шифр із застосуванням арифметичного апарату, розвиненого у попередньому параграфі. Користь такого підходу зрозуміла — обчислювальну техніку майже завжди простіше навчити оперувати із числовою інформацією, аніж із символьною. Основна домовленість, якої ми будемо дотримуватись до кінця цього параграфу, така:

n-символьний алфавіт утотожнюємо з кільцем $\mathbb{Z}_n$. А саме, кожна буква замінюється своїм номером у алфавіті, причому нумерація починається з нуля.

Наприклад, латинська абетка утотожнюється із $\mathbb{Z}_{26}$, а українська із $\mathbb{Z}_{33}$. Літера **а** української абетки трактується як 0, літера **б** як 1, **в** як 2 і т.д. Тепер до букв відкритого тексту ми можемо вільно застосовувати операції додавання та множення за відповідним модулем.

До кінця параграфу n буде служити позначенням для кількості букв у алфавіті відкритого тексту. Отже,

Шифр зсуву.

Ключ: s таке, що $0 \le s < n$.

Шифрування. У повідомленні кожна буква x заміщується буквою $E(x) = (x + s) \bmod n$.

Дешифрування. У криптотексті кожна буква x' заміщується буквою $D(x') = (x' + s') \bmod n$, де $s' = n - s$. Величину зворотнього зсуву s' будемо називати *дешифруючим ключем.*

За аналогією введемо у розгляд

Лінійний шифр.

Ключ: a таке, що $0 < a < n$ і НСД$(a, n) = 1$.

Шифрування. У повідомленні кожна буква x заміщується буквою $E(x) = (ax) \bmod n$.

Дешифрування. У криптотексті кожна буква x' заміщується буквою $D(x') = (a'x') \bmod n$, де $a' = a^{-1} \bmod n$ — дешифруючий ключ.

Приклад 3.1. Припустимо, повідомлення записуються українською абеткою без пропусків між словами та розділових знаків, тобто $n = 33$. Нехай для шифрування використовується ключ $a = 2$. За допомогою розширеного алгоритму Евкліда знаходимо, що $a' = 17$ (див. приклад 2.9). Розглянемо процедуру шифрування повідомлення **завтра**. У цифровій формі воно представляється послідовністю чисел 9,0,2,22,20,0. Множення на 2 за модулем 33 дає послідовність 18,0,4,11,7,0, яка відповідає криптотекстові **оагіеа**.

Дешифрування відбувається так само, лише із використанням дешифруючого ключа $a' = 17$. Наприклад, якщо ми маємо криптотекст **хдхт** = 25,5,25,22, то множення на 2 за модулем 33 приводить до 17,10,17,11 = **нині**.

Співвідношення $D(E(x)) = x$ для будь-якого $x \in \mathbb{Z}_n$ доводиться просто: $a'(ax) = (a'a)x = 1x = x$ (операції виконуються в $\mathbb{Z}_n$). Існування a' для a гарантоване умовою НСД$(a, n) = 1$ за твердженням 2.8. Більше того, a' для заданого a ефективно обчислюється за допомогою розширеного алгоритму Евкліда (приклад 2.9). Нарешті покажемо, що ті a, які не задовольняють накладену нами умову, непридатні для використання в якості ключа.

Твердження 3.2. *Відображення $E : \mathbb{Z}_n \to \mathbb{Z}_n$, задане формулою $E(x) = (ax) \bmod n$, має обернене тоді і тільки тоді, коли* НСД$(a, n) = 1$.

Доведення. В один бік твердження нами щойно було доведене. Навпаки, припустимо, що відображення E має обернене. За теоремою про обернене відображення (див. кінець пункту 1.1), E сюр'єктивне. Позначимо через e прообраз одиниці: $E(e) = 1$. Це означає, що $ae \equiv 1 \pmod n$, звідки й випливає, що НСД$(a, n) = 1$. ∎

Узагальненням і шифру зсуву, і лінійного шифру є

Афінний шифр.

Ключ: a, s такі, що $0 \le s < n$, $0 < a < n$ і НСД$(a, n) = 1$.

Шифрування. У повідомленні кожна буква x заміщується буквою $E(x) = (ax + s) \bmod n$.

Дешифрування. У криптотексті кожна буква x' заміщується буквою $D(x') = (a'x' + s') \bmod n$, де пара $a' = a^{-1} \bmod n$ і $s' = (-a's) \bmod n$ є дешифруючим ключем.

Приклад 3.3. Нехай для шифрування використовується ключ $a = 2$, $s = 1$. У прикладі 3.1 було обчислене значення $a' = 17$. Далі, $s' = (-17 \cdot 1) \bmod 33 = 16$. Щоб зашифрувати повідомлення **завтра**, представляємо його у цифровій формі як послідовність `9,0,2,22,20,0`. Множення на 2 за модулем 33 було виконане у прикладі 3.1. До результату додаємо 1 і отримуємо послідовність `19,1,5,12,8,1`, яка відповідає криптотекстові **пбдїжб**.

Дешифрування криптотексту **жсжд** $= 8, 21, 8, 5$ відбувається з використанням дешифруючого ключа $a' = 17$, $s' = 16$. Множення кожного із чисел за модулем 33 на 17 і додавання 16 дає $17, 10, 17, 11 =$ **нині**.

Як і кожен шифр простої заміни, афінний шифр піддається частотному аналізові. При цьому частотний метод використовується навіть не на повну потужність — див. вправи 3.3 і 3.4.

3.2. Афінні шифри вищих порядків. Подумаємо, як можна розширити монограмні шифри попереднього пункту так, щоб вони оперували з k-грамами для довільного $k > 1$. Спочатку введемо операцію додавання в $\mathbb{Z}_n^k$. Сумою векторів $X = (x_1, \ldots, x_k)$ і $S = (s_1, \ldots, s_k)$ з $\mathbb{Z}_n^k$ є вектор $X + S = ((x_1 + s_1) \bmod n, \ldots, (x_k + s_k) \bmod n)$. $\mathbb{Z}_n^k$ з операцією додавання є групою. Вектор $-S = (n - s_1, \ldots, n - s_k)$ є оберненим до вектора $S = (s_1, \ldots, s_k)$.

Шифр зсуву k-го порядку (шифр Віженера з періодом k).

Ключ: $S \in \mathbb{Z}_n^k$.

Шифрування. Повідомлення розбивається на k-грами. Кожна k-грама X заміщується k-грамою $E(X) = X + S$.

Дешифрування. Кожна k-грама X' криптотексту заміщується k-грамою $D(X') = X' + S'$, де $S' = -S$ є дешифруючим ключем.

Перед тим як перейти до лінійного шифру нагадаємо, що через $\mathrm{M}_k(\mathbb{Z}_n)$ ми позначаємо множину матриць розміру $k \times k$ з коефіцієнтами з кільця $\mathbb{Z}_n$, а через $\mathrm{GL}_k(\mathbb{Z}_n)$ — підмножину оборотних матриць. Для $A \in \mathrm{GL}_k(\mathbb{Z}_n)$ обернену до неї матрицю позначаємо через A^{-1}. Добутком AX матриці $A = (a_{ij})$ з $\mathrm{M}_k(\mathbb{Z}_n)$ на вектор-стовпчик $X = (x_1, \ldots, x_k)$ з $\mathbb{Z}_n^k$ є вектор-стовпчик

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1k} \\ a_{21} & a_{22} & \ldots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \ldots & a_{kk} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix} = \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1k}x_k \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2k}x_k \\ \vdots \\ a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{kk}x_k \end{pmatrix}$$

Лінійний шифр k-го порядку.

Ключ: $A \in \mathrm{GL}_k(\mathbb{Z}_n)$.

Шифрування. Повідомлення розбивається на k-грами. Кожна k-грама X заміщується k-грамою $E(X) = AX$.

Дешифрування. Кожна k-грама X' криптотексту заміщується k-грамою $D(X') = A'X'$, де $A' = A^{-1}$ — дешифруючий ключ.

Приклад 3.4. Лінійний шифр 1-го порядку обговорювався у попередньому пункті. Розглянемо докладніше випадок $k = 2$, тобто біграмний лінійний шифр. В якості ключа вибирається матриця $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ з коефіцієнтами $a, b, c, d \in \mathbb{Z}_n$. Матриця A повинна бути оборотною. За твердженням Б.5 це рівносильно умові $\mathrm{НСД}(w, n) = 1$ для $w = ad - bc$ — визначника матриці. За цієї умови з допомогою розширеного алгоритму Евкліда ми можемо знайти в $\mathbb{Z}_n$ обернений елемент w^{-1} і за формулою оберненої матриці обчислити дешифруючий ключ

$$A' = \begin{pmatrix} dw^{-1} \bmod n & -bw^{-1} \bmod n \\ -cw^{-1} \bmod n & aw^{-1} \bmod n \end{pmatrix}.$$

Наприклад, для $A = \begin{pmatrix} 1 & 1 \\ 32 & 1 \end{pmatrix}$ над $\mathbb{Z}_{33}$ маємо $w = 2$. За розширеним алгоритмом Евкліда знаходимо $w^{-1} = 17$ (див. приклад 2.9) і $A' = \begin{pmatrix} 17 & 16 \\ 17 & 17 \end{pmatrix}$.

Нехай потрібно зашифрувати повідомлення **завтра**. Першій біграмі **за** відповідає вектор $\begin{pmatrix} 9 \\ 0 \end{pmatrix}$. Множення на матрицю A дає $\begin{pmatrix} (1 \cdot 9 + 1 \cdot 0) \bmod 33 \\ (32 \cdot 9 + 1 \cdot 0) \bmod 33 \end{pmatrix} = \begin{pmatrix} 9 \\ 24 \end{pmatrix}$. Таким же чином знаходимо образи біграм **вт** $= \begin{pmatrix} 2 \\ 22 \end{pmatrix}$ та **ра** $= \begin{pmatrix} 20 \\ 0 \end{pmatrix}$. Зауважимо, що множення матриці A на три вектори-біграми еквівалентне множенню цієї матриці

на матрицю розміру 2 на 3:

$$\begin{pmatrix} 1 & 1 \\ 32 & 1 \end{pmatrix}\begin{pmatrix} 9 & 2 & 20 \\ 0 & 22 & 0 \end{pmatrix} = \begin{pmatrix} 9 & 24 & 20 \\ 24 & 20 & 13 \end{pmatrix}.$$

Насамкінець перетворюємо стовпчики отриманої матриці у біграми і отримуємо криптотекст **эффррй**.

Дешифрування відбувається так само, лише із використанням оберненої матриці. Наприклад, якщо ми маємо криптотекст **рьгк**, то розбиваємо його на біграми $\begin{pmatrix} \text{р} & \text{г} \\ \text{ь} & \text{к} \end{pmatrix} = \begin{pmatrix} 20 & 3 \\ 30 & 14 \end{pmatrix}$ і виконуємо множення на матрицю A':

$$\begin{pmatrix} 17 & 16 \\ 17 & 17 \end{pmatrix}\begin{pmatrix} 20 & 3 \\ 30 & 14 \end{pmatrix} = \begin{pmatrix} 17 & 17 \\ 10 & 11 \end{pmatrix} = \begin{pmatrix} \text{н} & \text{н} \\ \text{и} & \text{і} \end{pmatrix}.$$

В результаті дістаємо повідомлення **нині**.

Повернемось до загального аналізу лінійного шифру. Співвідношення $D(E(X)) = X$ для будь-якого $X \in \mathbb{Z}_n^k$ випливає з рівностей $A'(AX) = (A'A)X = I_kX = X$, де I_k — одинична матриця порядку k. Дешифруючий ключ A' для вибраної оборотною матриці A обчислюється ефективно за формулою для оберненої матриці (див. пункт Б.6). Потрібне для цього значення $(\det A)^{-1} \bmod n$ знаходиться за допомогою розширеного алгоритму Евкліда (приклад 2.9). На довершення доведемо, що необоротні матриці A непридатні для використання в якості ключа.

Твердження 3.5. *Відображення $E : \mathbb{Z}_n^k \to \mathbb{Z}_n^k$, задане формулою $E(X) = AX$, має обернене тоді і тільки тоді, коли $A \in \mathrm{GL}_k(\mathbb{Z}_n)$.*

Доведення. При $A \in \mathrm{GL}_k(\mathbb{Z}_n)$ існування відображення, оберненого до E, було встановлене вище. Навпаки, припустимо, що E має обернене відображення D. Позначимо через X_i, $1 \le i \le k$, вектор з i-тою компонентою 1 і з рештою компонент, що дорівнюють 0. Розглянемо матрицю U із стовпчиками $D(X_1), \ldots, D(X_k)$. Зауважимо, що $AU = I_k$, тобто матриця U є правою оберненою до A. Отже, матриця A оборотна (див. твердження Б.5). ∎

Тепер на черзі

Афінний шифр k-го порядку.

Ключ: $A \in \mathrm{GL}_k(\mathbb{Z}_n)$ і $S \in \mathbb{Z}_n^k$.

Шифрування. Повідомлення розбивається на k-грами. Кожна k-грама X заміщується k-грамою $E(X) = AX + S$.[1]

Дешифрування. Кожна k-грама X' криптотексту заміщується k-грамою $D(X') = A'X' + S'$, де $A' = A^{-1}$ і $S' = -A'S$ — дешифруючий ключ.

Приклад 3.6. Хочемо зашифрувати повідомлення **завтра** за допомогою ключа $A = \begin{pmatrix} 1 & 1 \\ 32 & 1 \end{pmatrix}$ і $S = \begin{pmatrix} 1 \\ 15 \end{pmatrix}$ над $\mathbb{Z}_{33}$.

Частина роботи вже виконана нами у прикладі 3.4. Після розбиття повідомлення на вектори-біграми і множення їх на матрицю A за модулем 33 ми були отримали $\begin{pmatrix} 9 & 24 & 20 \\ 24 & 20 & 13 \end{pmatrix}$. Додаємо до кожного стовпчика вектор S і отримуємо $\begin{pmatrix} 10 & 25 & 21 \\ 6 & 2 & 28 \end{pmatrix} = \begin{pmatrix} \text{и} & \text{х} & \text{с} \\ \text{е} & \text{в} & \text{ш} \end{pmatrix}$ тобто криптотекст **иехвсш**.

Знайдемо також дешифруючий ключ. У прикладі 3.4 була отримана матриця $A' = \begin{pmatrix} 17 & 16 \\ 17 & 17 \end{pmatrix}$. Обчислюємо $S' = -\begin{pmatrix} 17 & 16 \\ 17 & 17 \end{pmatrix}\begin{pmatrix} 1 \\ 15 \end{pmatrix} = \begin{pmatrix} 7 \\ 25 \end{pmatrix}$.

3.3. Криптоаналіз. *Атака з вибором відкритого тексту.* Нескладно зауважити, що афінний шифр нестійкий до цього виду криптоаналізу. Позначимо через X_i, $1 \le i \le k$, вектор з i-тою компонентою 1 і з рештою компонент, що дорівнюють 0, а через X_0 нульовий вектор. Супернику досить довідатись, в які криптотексти переходять відповідні цим векторам k-грами. Справді, $E(X_0) = S$, а образ $E(X_i)$ рівний i-тому стовпчику матриці A, що дозволяє повністю визначити ключ.

Атака з відомим відкритим текстом. Спочатку покажемо, що лінійний шифр вразливий від такої атаки. Припустимо суперник знає, що шифруюче відображення $E(X) = AX$ перетворює вектори $X_1, \ldots, X_k$ у вектори $X_1', \ldots, X_k'$. Сформуємо із стовпчиків $X_1, \ldots, X_k$

[1] Такими співвідношеннями в прямокутній системі координат задається кожне *афінне перетворення евклідового простору* $\mathbb{R}^k$. Прикметник *афінний* запозичено з грецької, де він означає *споріднений, подібний*. Це чудово узгоджується з тим фактом, що афінне перетворення переводить прямі у прямі і як наслідок зберігає багато геометричних властивостей. Однак вживання нами словосполучення *афінний шифр* є відходом від первинного значення, оскільки шифруюче відображення, навпаки, зобов'язане приховувати будь-яку спорідненість образу з оригіналом.

матрицю M, а із стовпчиків $X'_1, \ldots, X'_k$ матрицю C. Як неважко зрозуміти, $C = AM$ і $M = A'C$. Якщо матриця C оборотна, то звідси зразу можна визначити дешифруючий ключ $A' = MC^{-1}$. Якщо суперникові пощастить менше і матриця C виявиться необоротною, то він не зможе визначити A' однозначно. Однак кількість можливостей може зменшитись настільки, що A' вдасться знайти після деякого перебору.

Для афінного шифру $E(X) = AX + S$ необхідно знати на одну пару (X, X'), де $X' = E(X)$, більше. Віднявши рівність $AX + S = X'$ від кожної з рівностей $AX_i + S = X'_i$, $i \le k$, ми зведемо задачу визначення дешифруючого ключа до попереднього випадку.

Приклади такого криптоаналізу винесені у вправи 3.8–3.13.

Атака лише за криптотекстом. Афінний шифр 2-го порядку піддається частотному аналізу як і будь-який біграмний шифр. Якщо порядок k дещо збільшити, частотний метод перестане працювати.

Розглянемо лінійний шифр k-го порядку над алфавітом $\mathbb{Z}_n$. Подивимось, які перспективи може мати брутальна атака, іншими словами, наскільки реально влаштувати повний перебір ключів. Очевидно, кількість ключів дорівнює кількості матриць в $\mathrm{GL}_k(\mathbb{Z}_n)$, для якої в пункті 2.5 було введено позначення $\phi_k(n)$. Там же була виведена формула $\phi_k(n) = n^{k^2} \prod_{p|n} \prod_{i=1}^{k} (1 - 1/p^i)$. З одного боку видно, що $\phi_k(n) \le n^{k^2}$. З іншого, порівнюючи із формулою для функції Ойлера $\phi(n) = \phi_1(n)$, отримуємо $\phi_k(n) \ge n^{k^2-k}(\phi(n))^k$. З використанням твердження 2.15 маємо оцінку $\phi_k(n) \ge n^{k^2}/(6 \ln \ln n)^k$ при $n > 4$. Асимптотично, значення $\phi_k(n)$ зростає за n як поліном і за k як експоненційна функція. Отримана оцінка дозволяє оцінити розмір простору ключів для будь-яких конкретних k і n (див. також вправу 3.20). Як видно, перебір ключів напевне є нереальним скажімо при $k = 5$, $n = 32$. Платою за збільшення порядку шифру є збільшення часу криптування і декриптування.

ВПРАВИ

3.1. Для монограмного афінного шифру перевірити співвідношення $D(E(x)) = x$, де x — довільний елемент із $\mathbb{Z}_n$.

3.2. a) Зашифрувати за допомогою афінного шифру з ключем $a = 4$, $s = 3$ повідомлення `ні`, записане українською абеткою із 33 літер.

b) Знайти дешифруючий ключ і розшифрувати криптотекст `дкцзьи`, отриманий за допомогою того ж шифру з тим же ключем.

3.3. a) Каналом зв'язку передаються повідомлення, закриптовані за допомогою монограмного лінійного шифру, причому використовується 35-

символьний алфавіт, у якому під номерами від 0 до 32 йдуть літери української абетки, пропуск має номер 33, а крапка — 34. Частотний аналіз показав, що у потоці криптотекстів найчастіше зустрічається літера `Щ`. Опираючись на факт, що найпоширенішим символом в україномовних текстах є пропуск, знайти дешифруючий ключ і розшифрувати криптотекст

`ТЬЕПЩЧАЕОЧИЬЯЩЕ␣ЛЕАМЮАФ.ҐЮОЩХАЄ␣ЕЯ`

Знайти шифруючий ключ і закриптувати повідомлення

`КРЕВЕТКИ␣ЗАКІНЧИЛИСЬ.`

b) Повідомлення криптуються з використанням того ж шифру, але використовується 33-символьний алфавіт, в якому лише літери української абетки. Згідно з результатами частотного аналізу, у потоці криптотекстів найчастіше зустрічається літера `Ф`. В числі інших перехоплено криптотекст `УФЇЄФГШЖАЙХФ`. Розшифрувати його, опираючись на факт, що найпоширенішою літерою в україномовних текстах є `О`. Знайти шифруючий ключ і закриптувати повідомлення `ЗАВЕРШУЙТЕ`.

c) Використовується той же шифр, але над 36-символьним алфавітом. Першими символами алфавіту є крапка, кома та пропуск, які мають номери 0, 1 та 2 відповідно. Номери від 3 до 35 належать літерам української абетки. Відомо, що у потоці криптотекстів найчастіше зустрічаються літери `Є` і `Ь`, саме в такому порядку. Виходячи з того, що найпоширенішими в україномовних текстах є пропуск і літера `О`, знайти дешифруючий ключ і розшифрувати криптотекст `ЬЛЮЗЇРЄИЄҐЯЇВЖҐЗИЄҐФШЬЯЖ.ЄНЇН`. Знайти шифруючий ключ і закриптувати повідомлення `ЗАБУДЬ␣УСЕ,␣БОБЕ`.

3.4. a) Каналом зв'язку передаються повідомлення, написані у 33-літерному українському алфавіті без пропусків і знаків пунктуації, і закриптовані за допомогою монограмного афінного шифру. Відомо, що у потоці криптотекстів найчастіше зустрічаються літери `У` і `О`, саме в такому порядку. Виходячи з того, що найпоширенішими літерами української абетки є `О` і `Н`, знайти дешифруючий ключ і розшифрувати криптотекст `ЗУКУОН`. Знайти шифруючий ключ і закриптувати повідомлення `ЗАГАСИТИ`.

b) Повідомлення написані англійською мовою у 26-літерному алфавіті без пропусків і розділових знаків, і закриптовані за допомогою монограмного афінного шифру. Відомо, що у потоці криптотекстів найчастіше зустрічаються літери `O` і `H`, саме в такому порядку. Виходячи з того, що найпоширенішими літерами англійської абетки є `E` і `T`, знайти дешифруючий ключ і розшифрувати криптотекст `HSFOSBPSHHSFO`. Знайти шифруючий ключ і закриптувати повідомлення `NOQUESTIONS`.

3.5. a) Довести, що афінне відображення однозначно представляється у вигляді композиції лінійного відображення та зсуву.

b) Довести, що афінний шифр утворює групу (кількість букв у алфавіті відкритого тексту n фіксована).

c) Знайти порядок цієї групи як функцію від n. Обчислити його при $n = 26, 33$.

d) Довести, що лінійні шифри і шифри зсуву утворюють у цій групі підгрупи. Чи є ці підгрупи нормальними?

3.6. *Нерухомою буквою* відносно шифруючого відображення E назвемо букву x із властивістю $E(x) = x$. Скільки букв залишає нерухомими відображення $E(x) = (ax + s) \bmod n$?

3.7. a) До матриці $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ знайти обернену в $\mathbb{Z}_{33}$.

b) Використовуючи матрицю A як ключ, закриптувати за допомогою лінійного шифру 2-го порядку повідомлення ПРОЩАЙ .

c) Використовуючи в якості ключа ту ж матрицю A і вектор $S = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$, закриптувати те ж повідомлення за допомогою афінного шифру.

d) Знайти дешифруючий ключ і дешифрувати криптотекст ЛЩЯЛЧОРХ, отриманий за допомогою афінного шифру з ключем як у попередньому пункті.

3.8. a) Суперник знає, що Аліса та Боб листуються українською мовою і криптують свою кореспонденцію за допомогою лінійного шифру 2-го порядку. При цьому використовується 34-символьний алфавіт, де номери від 0 до 32 належать літерам української абетки, а 33-тім символом є пропуск. Суперникові вдалося підслухати повідомлення Аліси Бобові: ҐТІШГҐРЮІДЖСМАЧДИКЯ␣ЄЦЖС . Він здогадався, що останнім словом у повідомленні є підпис відправника АЛІСА. Виходячи з цього припущення, знайти дешифруючий ключ і розшифрувати криптотекст. Знайти шифруючий ключ і закриптувати повідомлення ЧЕКАЮ␣БІЛЯ␣ФОНТАНУ␣БОБ

b) У тій же ситуації, що й в попередньому пункті, суперник перехопив криптотекст ДТЛРНІЇРЦДЄЙМЗЧОТШБЕ. Цього разу суперник здогадався, що повідомлення починається звертанням БОБЕ. Розшифрувати криптотекст, виходячи з цього припущення. Знайти шифруючий ключ і закриптувати таке ж, як і в попередньому пункті, повідомлення ЧЕКАЮ␣БІЛЯ␣ФОНТАНУ␣БОБ

3.9. a) Відомо, що використовується біграмний лінійний шифр над 33-літерним українським алфавітом, занумерованим числами від 0 до 32. Пропуски між словами ігноруються. Статистичний аналіз показав, що в потоці криптотекстів найчастіше зустрічаються біграми НЮ і ПБ. Виходячи з припущення, що в україномовних текстах з предмету, про який йдеться у повідомленні, найпоширенішими є біграми СТ і НА, знайти дешифруючий ключ і розшифрувати повідомлення НЮЛВПБИІДИЧТ

b) [111] Використовується 34-символьний алфавіт, в якому 33-ом літерам російської абетки відповідають номери 0–32, а пропуск має номер 33. Статистичний аналіз показав, що в потоці криптотекстів найчастіше зустрічаються

біграми ЮТ і ЧМ. Виходячи з припущення, що вони відповідають біграмам НО і ЕТ, найпоширенішим у російськомовних текстах з предмету листування, прочитати підслухане повідомлення СХНСЪШОНЩЗ

c) Використовується 26-літерний англійський алфавіт, занумерований числами від 0 до 25. Пропуски між словами ігноруються. Статистичний аналіз показав, що в потоці криптотекстів найчастіше зустрічаються біграми VO і IT. Припустимо, що в англомовних текстах на тему, яка обговорюється в повідомленнях, найчастіше зустрічаються біграми TH і HE. Знайти дешифруючий ключ і розшифрувати повідомлення ITEJASVOQOXT

3.10. a) Повідомлення шифруються лінійним біграмним шифром над 30-символьним алфавітом, в якому номери від 0 до 25 займає латинська абетка, а пропуск, апостроф, кома і крапка, саме в такому порядку, мають номери 26–29. Статистичний аналіз великого масиву криптотекстів показав, що найчастіше трапляються біграми EI і QQ. Припустимо, що вони відповідають найпоширенішим в англійській мові біграмам E␣ і S␣ цього алфавіту. Встановити дешифруючий ключ і розшифрувати криптотекст LHV,QQQW␣HESLSEIWYRVGYQ␣BRBC

b) Відомий розвідник користується біграмним лінійним шифром над 34-символьним алфавітом, у який входять українські літери (0–32) і пропуск (33). Втім, щоб ускладнити криптоаналіз, досвідчений розвідник при шифруванні ігнорує всі пропуски між словами (таким чином, у відкритому тексті немає пропусків, але у криптотексті вони можуть з'явитись). Суперник перехопив повідомлення ОЩРФНААИЗЖЕБ␣ЗПЗ розвідника в центр і здогадався, що останні п'ять символів криптотексту відповідають підпису відправника ІСАЄВ. Провести дешифрування.

3.11. a) Перехоплено криптотекст ЮВЧРУЗІНДШЛЗТАЬВБЯІТЬГКЇ , отриманий за допомогою лінійного біграмного шифру над 33-літерним алфавітом (пропуски між словами ігноруються). Відомо, що повідомлення закінчується підписом відправника НАТАЛКА. Знайти дешифруючий ключ і прочитати повідомлення.

b) Перехоплено криптотекст ШЩЄВЛОІІЩШАФУАРЇАБННЕЮ , отриманий за допомогою лінійного біграмного шифру над 34-символьним алфавітом, у який входить українська абетка (0–33) і пропуск (33). Відомо, що повідомлення закінчується підписом відправника ␣ВАНГА. Знайти дешифруючий ключ і прочитати повідомлення.

3.12. Каналом зв'язку передаються повідомлення, закриптовані за допомогою афінного біграмного шифру над 34-символьним алфавітом, у який входить українська абетка (0–33) і пропуск (33). Статистичний аналіз виявив, що найчастіше в потоці криптотекстів зустрічаються біграми РТ, ГД і ІВ. Виходячи з припущення, що вони відповідають найпоширенішим в українській мові біграмам ␣П, ␣В і И␣ цього алфавіту, знайти дешифруючий ключ A', S' і розшифрувати перехоплений криптотекст ЛЖІВФЗХЗРТОЖГДШУССЯВПГЗДБЬ␣У

3.13. (За [111]) Перехоплено повідомлення

ЕЬЩИЦЕППІДНПТЛЧТХХИШХКПСҐТНВТУ

отримане за допомогою лінійного шифру 3-го порядку над 33-літерним алфавітом (пропуски між словами ігноруються). Повідомлення закінчується підписом відправника ДЖЕЙМСБОНД, що дає можливість встановити відповідність між трьома триграмами повідомлення і криптотексту. Знайти дешифруючий ключ і прочитати повідомлення.

3.14. a) Для лінійного шифру k-го порядку над алфавітом $\mathbb{Z}_n$ довести, що композиція шифруючих відображень з ключами $A, B \in \mathrm{M}_k(\mathbb{Z}_n)^*$ (спочатку шифрування з ключем A, а потім з B) також є шифруючим відображенням з ключем BA.

b) Довести, що лінійний шифр k-го порядку над алфавітом $\mathbb{Z}_n$ утворює групу, ізоморфну групі $\mathrm{M}_k(\mathbb{Z}_n)^*$.

3.15. Виконати вправу 3.5 для афінного шифру k-го порядку. Чисельні підрахунки у пункті **c** провести при $k = 2$.

3.16. Розглянути лінійне шифруюче відображення $E : \mathbb{Z}_n^k \to \mathbb{Z}_n^k$, задане формулою $E(X) = AX$ з $A \notin \mathrm{M}_k(\mathbb{Z}_n)^*$. Довести, що

a) кожна біграма з $E(\mathbb{Z}_n^k)$ має щонайменше два прообрази;

b) кожен криптотекст, отриманий шифруванням деякого повідомлення довжини km, можна розшифрувати щонайменше 2^m способами.

3.17. ([111]) Нехай неодинична матриця A з $\mathrm{M}_2(\mathbb{Z}_n)$ використовується як ключ для лінійного біграмного шифру E в n-символьному алфавіті. *Нерухомою біграмою* шифруючого відображення E назвемо таку біграму (вектор) X, що $E(X) = X$.

a) Знайти умову на матрицю A, за якої існує єдина нерухома біграма, а саме aa $= \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, де a — перша буква алфавіту.

b) Припустимо n просте і aa — не єдина нерухома біграма. Довести, що в цьому разі є рівно n нерухомих біграм.

3.18. *Інволюцією* називається відображення множини на себе, яке є оберненим саме до себе. Зрозуміло, що для шифруючого відображення бути інволюцією означає збігатися з дешифруючим відображенням. Для лінійного шифру шифруюче відображення, що задається ключем A, є інволюцією тоді і тільки тоді, коли матриця A обернена сама до себе. Таку матрицю будемо називати *матрицею інволюції*.

a) Довести, що визначник матриці інволюції $A \in \mathrm{M}_k(\mathbb{Z}_p)^*$, де p просте, дорівнює 1 або -1.

b) Порахувати кількість матриць інволюцій $A \in \mathrm{M}_2(\mathbb{Z}_p)^*$ з $\det A = 1$, де p просте.

c) Порахувати кількість матриць інволюцій $A \in \mathrm{M}_2(\mathbb{Z}_p)^*$ з $\det A = -1$, де $p > 2$ просте.

d) Скільки є матриць інволюцій в $\mathrm{M}_2(\mathbb{Z}_{33})^*$?

e) ([3]) Скільки є матриць інволюцій в $\mathrm{M}_2(\mathbb{Z}_{26})^*$?

f) Виписати всі матриці інволюцій в $\mathrm{M}_2(\mathbb{Z}_{26})^*$ з визначником 1.

3.19. a) Довести, що шифр перестановки періоду k можна реалізувати за допомогою лінійного шифру k-го порядку, тобто, для кожного ключа σ шифру перестановки існує ключ A лінійного шифру такий, що σ і A задають одне й те ж шифруюче відображення.

b) Нехай $k = 5$. Знайти A для $\sigma = (135)(24)$.

3.20. Розглянемо лінійний шифр k-го порядку над алфавітом $\mathbb{Z}_2$ (повідомлення записується у двійковій формі і шифрується блоками довжини k — такий шифр називається *збовтуючим*). Простором ключів цього шифру є множина $\mathrm{M}_k(\mathbb{Z}_2)^*$. Задамось питанням, наскільки реально вибрати *випадковий* ключ.

a) Довести, що для будь-якого k ймовірність того, що випадкова матриця з $\mathrm{M}_k(\mathbb{Z}_2)$ є оборотною, перевищує $1/4$.

b) Розглянемо таку процедуру знаходження оборотної матриці порядку k над $\mathbb{Z}_2$. Кожен крок процедури полягає у виборі випадкової матриці і обчисленні її визначника. Якщо він виявився рівним 1, то оборотну матрицю знайдено і процедура завершується, якщо ж визначник виявився рівним 0, то виконується наступний крок. Виходячи з пункту a, оцінити зверху математичне сподівання кількості кроків, необхідних для успішного знаходження оборотної матриці.

c) Виходячи з пункту a, оцінити знизу ймовірність того, що серед 10 матриць, вибраних випадково і незалежно між собою, знайдеться принаймні одна оборотна.

ЛІТЕРАТУРА

Системний розгляд афінних шифрів належить Гіллові [105]. У нашому викладі цього предмету використовується досвід Ніла Кобліца [111, розділ III].

Розділ III.

Алгоритми та їх складність

Програмістська практика добре розрізняє алгоритми швидкі та повільні. Ці характеристики вкрай важливі для криптографії. З одного боку, користувачі криптосистеми повинні володіти швидкими алгоритмами шифрування та дешифрування, а з іншого — їх суперник не повинен мати швидкого алгоритму розкриття криптотексту без знання ключа. В ідеалі, суперник не має швидкого алгоритму розкриття не тому, що йому бракує хисту додуматись до такого алгоритму, а тому, що останнього взагалі не існує — всі алгоритми зламу криптосистеми є повільними.

Наявність швидкого алгоритму для розв'язання певної задачі доводиться безпосереднім пред'явленням такого алгоритму. При цьому алгоритм може бути заданим неформальним інтуїтивно прийнятним описом, який, як правило, нескладно записати як програму у будь-якій з існуючих мов програмування. Відсутність же швидкого алгоритму для задачі є фактом більш фундаментального характеру, доведення якого вимагає глибоких математичних методів. Галуззю математики, що займається подібними питаннями, є *теорія складності*. Цей розділ нашого курсу присвячений концепціям теорії складності, на яких буде грунтуватись подальший виклад алгоритмічних задач теорії чисел, що на них базується сучасна криптографія.

§ 1. Поняття та терміни

1.1. Задачі. Ми розглядатимемо функції вигляду $f : \mathcal{A}^* \to \mathcal{B}^*$, де $\mathcal{A}$ і $\mathcal{B}$ — деякі алфавіти. Саме цей клас функцій виникає, коли ми цікавимось функціями, які можна ефективно обчислювати. Наприклад,

функцію піднесення до квадрату у множині невід'ємних цілих чисел з обчислювальної точки зору природньо трактувати як функцію із множини $\{0,1,\dots,9\}^*$ в себе. Слід лише домовитись, як бути з десятковими словами, що починаються цифрою 0. Їх можна утотожнити з невід'ємними цілими десятковими числами, що отримуються після відкидання перших нулів. А можна вважати, що функція відображає слова, які не є десятковим записом натурального числа, у 0. Якщо функція двомісна, на зразок додавання натуральних чисел, то для розділення аргументів зручно ввести в алфавіт новий спеціальний символ, як от #.

Задача обчислення функції $f : \mathcal{A}^* \to \mathcal{B}^*$ полягає у знаходженні для вказаного слова $w \in \mathcal{A}^*$ значення функції $f(w)$. Ми будемо описувати задачі обчислення наступним чином:

Задано: $w \in \mathcal{A}^*$.
Обчислити: $f(w)$.

Втім, коли нам не буде настільки важливо, в якому алфавіті і як саме кодуються аргументи та значення функції, ми допускатимемо і менш формальні описи задач на кшталт

Задано: $x \in \mathbb{N}$.
Обчислити: x^2.

Іноді задачу у вищеозначеному сенсі ми будемо називати *масовою*. Конкретне задане $w \in \mathcal{A}^*$ називатимемо *індивідуальною* задачею. Значення $f(w)$ будемо називати *розв'язком* індивідуальної задачі w. Масову задачу можна вважати нескінченним набором індивідуальних задач.

Нехай $L \subseteq \mathcal{A}^*$ — множина слів у алфавіті $\mathcal{A}^*$. Часто L називають *мовою*. *Задача розпізнавання* мови L полягає у визначенні, належить задане слово $w \in \mathcal{A}^*$ цій мові, чи ні:

Задано: $w \in \mathcal{A}^*$.
Розпізнати: $w \in L$?

Наприклад,

Задано: $x \in \mathbb{N}$.
Розпізнати: чи є x повним квадратом?

Індивідуальною задачею є будь-яке задання слова w. Розв'язком індивідуальної задачі є відповідь "так" чи "ні", яку прийнято кодувати символами 1 та 0 відповідно.

Ще один різновид задачі, який нам буде траплятися, називається *задачею пошуку* елемента із заданою властивістю. Нехай алфавіт $\mathcal{A}$ не містить символу # і $P \subseteq (\mathcal{A} \cup \{\#\})^*$. Для кожного заданого $w \in \mathcal{A}^*$ задача полягає або у знаходженні хоча б одного $u \in \mathcal{A}^*$ такого, що $w\#u \in P$, або у констатації, що елемента u з такою властивістю немає:

Задано: $w \in \mathcal{A}^*$.
Знайти: u *таке, що* $w\#u \in P$.

Індивідуальною задачею є довільно задане слово $w \in \mathcal{A}^*$, а її розв'язком є або відповідне u, або відповідь "не існує", яку зручно кодувати символом #. Наприклад,

Задано: $x \in \mathbb{N}$.
Знайти: $y \in \mathbb{Z}$ *таке, що* $x = y^2$.

Наведений приклад задачі пошуку подано у не до кінця формалізованій формі — деталі формалізації, включно з вибором алфавіту $\mathcal{A}$, залишені читачеві в якості простої вправи.

Ми ввели три типи задач — обчислення, розпізнавання та пошуку. Насправді, в обчислювальному відношенні всі вони рівносильні між собою — кожну задачу одного типу можна переформулювати як задачу будь-якого з двох інших типів.

Так, задача розпізнавання мови $L \subseteq \mathcal{A}^*$ є ні чим іншим, як задачею обчислення її *характеристичної функції* $\chi_L : \mathcal{A}^* \to \{0,1\}$, що набуває значення 1 на аргументах з L і лише на них. Задачу обчислення функції $f : \mathcal{A}^* \to \mathcal{B}^*$ легко подати як задачу пошуку, взявши $P = \{\, w\#f(w) : w \in \mathcal{A}^* \,\}$, де P — множина слів у алфавіті $\mathcal{A} \cup \mathcal{B} \cup \{\#\}$. Нарешті, кожну задачу пошуку можна звести до деякої задачі розпізнавання (див. приклад 4.1).

1.2. Алгоритми. Поняття алгоритму в математиці таке ж давнє і фундаментальне, як, скажімо, поняття числа. Наприклад, алгоритмові Евкліда не менш як 22 сотні років. Подібно до того, як розвивалось уявлення про дійсні числа, від відкриття давньогрецькими математиками неспівмірності довжин сторони квадрата і його діагоналі і аж до чітких конструкцій та означень, розроблених у другій половині 19 століття Г. Кантором, Ш. Мере, Р. Дедекіндом та К. Вайерштрасом, поняття алгоритму теж уточнювалось і було строго означене у першій половині нашого століття. Перші формалізації запропонували у 1936 році Е. Пост і А. Тьюрінг, випередивши появу обчислювальної техніки, яка значною мірою грунтується на тій же ідеології. Згодом були запропоновані й інші означення, зокрема А. А. Марковим та А. М. Колмогоровим, і всі вони виявились еквівалентними між собою. Формалізація поняття алгоритму є видатним досягненням математичної логіки, оскільки лише з появою строгого означення алгоритму стало можливим *доводити* для певних задач, що *жоден* алгоритм не здатен їх розв'язати. Такі задачі називаються *алгоритмічно нерозв'язними*. Так, задача розпізнавання, чи заданий многочлен від кількох змінних з цілими коефіцієнтами має цілі нулі, є алгоритмічно нерозв'язною (десята проблема Гільберта). Зазначимо для

порівняння, що у випадку многочленів від однієї змінної для подібної задачі алгоритм відомий.

З огляду на вступний характер цього посібника, ми не вводимо формального означення алгоритму. На виправдання такого розвантаження викладу можна зауважити, що

- ніщо в нашому курсі не залежить від вибору одного з багатьох можливих означень алгоритму;
- для розуміння предмету цілком достатнім є інтуїтивне уявлення про процес обчислення як про роботу ЕОМ згідно із програмою, написаною на читача улюбленій мові програмування;
- зацікавлений читач знайде весь доречний формалізм у підручниках [7, 18].

Коли йтиметься про *алгоритм*, то вважатимуться зафіксованими два алфавіти — *вхідний* $\mathcal{A}$ та *вихідний* $\mathcal{B}$. Робота алгоритму полягає в тому, що він *отримує на вхід* слово у вхідному алфавіті — *вхід*, і як результат виконання послідовності *елементарних операцій*, *подає на вихід* слово у вихідному алфавіті — *вихід*. Поняття елементарної операції або *кроку роботи* є складовою формального означення алгоритму. Ми не будемо уточнювати це поняття, апелюючи до його побутово-програмістського розуміння.

Алгоритм *розв'язує масову задачу*, якщо, отримавши на вхід будь-яку індивідуальну задачу $w \in \mathcal{A}^*$, він за скінченну кількість кроків подає на вихід її розв'язок. Залежно від типу задачі говоримо, що алгоритм *обчислює* функцію, *розпізнає* множину чи мову, *знаходить* елемент із певною властивістю. У випадку задачі розпізнавання, якщо алгоритм подає на вихід 1, то кажемо, що він *приймає вхід*, а якщо 0, то кажемо, що алгоритм *відхиляє вхід*.

Довжиною входу $w \in \mathcal{A}^*$ називається кількість букв у слові w, яку ми домовились позначати як $|w|$. Нехай $t : \mathbb{N} \to \mathbb{N}$ — деяка функція. Алгоритм *розв'язує задачу за час* t, якщо на кожному вході w він робить не більше, ніж $t(|w|)$ кроків. Якщо $t(n) \le cn^c$ для деякої константи c, то алгоритм розв'язує задачу *за поліноміальний від довжини входу час*. Такий алгоритм називають *поліноміальним*, а задачу *поліноміально розв'язною*.

Поняття поліноміального часу є центральною концепцією теорії складності обчислень. В рамках цієї концепції вважається, що поліноміальні алгоритми відповідають *швидким, ефективним* на практиці алгоритмам; а поліноміально розв'язні задачі відповідають *легким* задачам, які можуть бути розв'язаними за прийнятний час на входах

довжини, що має практичний інтерес. Часто поліноміальний алгоритм справді задовольняє всі практичні проблеми. В гіршому ж випадку його можна вважати швидким лише асимптотично, а на відносно невеликих входах алгоритм може працювати довго.

Слід підкреслити, що клас поліноміально розв'язних задач не залежить від того, яка саме з багатьох можливих формалізацій алгоритму вибрана. Цей факт можна строго довести як теорему для будь-яких двох означень алгоритму.

Клас поліноміально розв'язних задач найчастіше не залежить і від способу, в який задача сформульована. Наприклад, задача множення натуральних чисел ефективно розв'язується як у двійковій, так і в десятковій системах числення. Використання іншого алфавіту не здатне суттєво прискорити обчислення (тобто зменшити час як функцію від довжини входу). Винятком є формулювання задачі в односимвольному алфавіті. Ми виключаємо цей випадок з нашого розгляду як такий, що не має застосувань (див. вправи 1.4–1.6).

Алгоритм, який на нескінченній послідовності входів робить більше як 2^{n^c} кроків, де n довжина входу, а $c > 0$ — деяка константа, називається експоненційним. Про такий алгоритм кажуть, що він вимагає *експоненційного часу*. Експоненційні алгоритми відповідають загальним уявленням про *повільні, неефективні* на практиці алгоритми. Як приклад згадаємо алгоритм ламання шифру повним перебором ключів. Задачу, яка може бути розв'язана лише експоненційним алгоритмом, слід визнати *важкою*. Типовою є ситуація, коли для задачі відомі лише експоненційні алгоритми, але не вдається довести, що кращих не існує. На практиці така задача теж вважається важкою (до моменту, поки для неї не буде знайдено ефективного алгоритму, якщо такий все ж виявиться). Як приклад наведемо задачу пошуку, на важкості якої грунтується чимало сучасних криптосистем:

Задано: $x \in \mathbb{N}$.
Знайти: *нетривіальний дільник числа* x.

Варто уточнити, що експоненційний алгоритм є повільним *у найгіршому випадку* — досить, щоб він затрачав час 2^{n^c} хоча б на одному з k^n входів довжини n, де k кількість букв у вхідному алфавіті. Може трапитись, що повільний в найгіршому випадку алгоритм є досить швидким *у середньому*, тобто для переважної більшості входів довжини n. (Зауважимо, що можуть розглядатися різні ймовірнісні розподіли на множині входів довжини n — рівномірний розподіл не завжди найприродніший.) Хоч і експоненційний, але швидкий у середньому алгоритм може бути цілком придатним для практичних потреб. Показовим прикладом є *симплекс-метод* для задачі *лінійного програмування*. Наявність алгоритмів, які добре працюють в середньому, слід враховувати при оцінці складності задачі.

Нагадаємо, що множина індивідуальних задач є множиною всіх слів у деякому алфавіті $\mathcal{A}$. Інколи є сенс розглядати *звуження* масової задачі на деяку підмножину індивідуальних задач $S \subset \mathcal{A}^*$. Алгоритм розв'язує звужену задачу, якщо він видає правильний розв'язок для індивідуальних задач із множини S (а на всіх інших входах може подавати на вихід будь-що, або й взагалі не зупинятись).

ВПРАВИ

1.1. Подати функції, що пропонуються, у формі $f : \mathcal{A}^* \to \mathcal{B}^*$ для деяких алфавітів $\mathcal{A}$ і $\mathcal{B}$:

a) f ставить у відповідність парі раціональних чисел частку від ділення першого на друге;

b) f ставить у відповідність скінченній множині натуральних чисел кількість її елементів.

1.2. У рамках неформального поняття поліноміального часу довести, що якщо задачі розпізнавання мов L_1 і L_2 поліноміально розв'язні, то такими ж є задачі розпізнавання мов $L_1 \cap L_2$ і $L_1 \cup L_2$.

1.3. Придумати ін'єктивні відображення $f_1 : \{0,\dots,9\}^* \to \{0,1\}^*$ і $f_2 : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$, які обчислюються швидкими алгоритмами.

1.4. Описати алгоритми обчислення функцій $f_1 : \{0,\dots,9\}^* \to \{0,1\}^*$ і $f_2 : \{0,1\}^* \to \{0,\dots,9\}^*$, які переводять десятковий запис невід'ємного цілого числа у двійковий і навпаки.

1.5. Розглянемо функцію $f : \mathcal{A}^* \to (\mathcal{A} \cup \{\#\})^*$, яка натуральному числу x ставить у відповідність $1\#2\#\dots\#x$, список всіх натуральних чисел, що не перевищують x.

a) Нехай $\mathcal{A} = \{0,1\}$ і натуральні числа задаються у двійковій системі. Довести, що будь-який алгоритм для обчислення f повинен зробити на вході довжини n не менше, ніж 2^n елементарних операцій. (*Вказівка:* запис кожного символу виходу вимагає щонайменше однієї елементарної операції.)

b) Нехай $\mathcal{A} = \{1\}$ і натуральні числа задаються в унарній системі, тобто $x = 11\dots1$ (x цифр). Описати алгоритм обчислення функції f, який на вході довжини n робить $O(n^2)$ кроків.

1.6. Довести, що жодне ін'єктивне відображення $f : \{0,1\}^* \to \{1\}^*$ не може бути обчислене за поліноміальний час.

ЛІТЕРАТУРА

Джерелами з класичної теорії алгоритмів є, серед інших, [28, 40, 50], а з теорії складності — підручники [7, 18] та огляд [54].

§ 2. Прямолінійні програми

Цей параграф присвячений класові алгоритмів, які називаються *прямолінійними програмами*. Для нас важливим є те, що деякі необхідні для криптографічної практики алгоритми природно описувати саме прямолінійними програмами. Читач, налаштований суто прагматично, повинен зосередитись на *бінарному алгоритмі* піднесення до степеня з пункту 2.2. Пункт 2.3 призначений для читача з ширшими інтересами, що включають теоретичні основи сучасного криптоаналізу.

2.1. Означення моделі. Поняття прямолінійної програми можна вводити для довільної алгебраїчної структури. Ми зробимо це для кільця з одиницею. Елементарними операціями будуть множення та додавання в кільці.

Нехай задані наступні об'єкти.

- Символ 1, який називається *символом предметної константи* і позначає одиницю кільця.
- Алфавіт $\mathcal{X} = \{x_1, \ldots, x_m\}$, елементи якого називаються *вхідними змінними*.
- Алфавіт $\mathcal{Z} = \{z_1, \ldots, z_l\}$, елементи якого називаються *службовими змінними*.
- Три *символи операцій* $\cdot, +, -$, для множення, додавання та віднімання.

Прямолінійною програмою називається послідовність із l слів у алфавіті $\mathcal{X} \cup \mathcal{Z} \cup \{1, \cdot, +, -, =\}$, в якій i-те слово має вигляд $z_i = z' \circ z''$, де $\circ$ є одним із символів операцій, а z', як і z'', є або символом предметної константи, або вхідною змінною, або однією із службових змінних $z_1, \ldots, z_{i-1}$.

Слова, з яких складається прямолінійна програма, називаються її *командами*. Зауважимо, що кожна команда виконує операцію над операндами, які можуть бути лише результатами виконання попередніх команд. Кількість команд прямолінійної програми, яку ми позначили через l, називається її *довжиною* або *складністю*. Кількість команд множення називається *мультиплікативною складністю*, а додавання та віднімання — *адитивною складністю* програми.

Припустимо, що в нас є прямолінійна програма, і нехай R — деяке кільце з одиницею. Кожна команда прямолінійної програми визначає деяку функцію $f : R^m \to R$, яка є поліномом від змінних $x_1, \ldots, x_m$. Справді, перша команда задає поліном вигляду 1, $x_j \pm 1$, $x_j \pm x_i$, або $x_j \cdot x_i$. Кожна наступна команда задає функцію, що є добутком, сумою або різницею функцій, визначених якимись із попередніх команд.

Функцію $F : R^m \to R^k$ розглядаємо як набір k функцій-проекцій $f_j : R^m \to R$, де $F(x_1, \ldots, x_m) = f_1(x_1, \ldots, x_m), \ldots, f_k(x_1, \ldots, x_m)$. Прямолінійна програма *обчислює* функцію $F : R^m \to R^k$, якщо кожна з функцій f_j, де $1 \le j \le k$, визначається деякою командою цієї програми. *Складністю обчислення функції F прямолінійними програмами* називається найменша довжина прямолінійної програми, що обчислює F. Подібно означуються мультиплікативна та адитивна складності функції.

Приклад 2.1. Функція $F(x) = x^4 + x^2$ в будь-якому кільці обчислюється такою програмою:

$$\begin{aligned} z_1 &= x \cdot x \\ z_2 &= z_1 \cdot x \\ z_3 &= z_2 \cdot x \\ z_4 &= z_3 + z_1. \end{aligned}$$

Отже, складність цієї функції не перевищує 4. Насправді вона менша, тому що для обчислення функції є коротша програма:

$$\begin{aligned} z_1 &= x \cdot x \\ z_2 &= z_1 + 1 \\ z_3 &= z_1 \cdot z_2. \end{aligned}$$

2.2. Піднесення до степеня. Розглянемо задачу обчислення функції $f(x) = x^d$ у довільному кільці. Безхитрісна прямолінійна програма для цієї функції має складність $d - 1$:

$$\begin{aligned} z_1 &= x \cdot x \\ z_2 &= z_1 \cdot x \\ z_3 &= z_2 \cdot x \\ &\ldots \\ z_{d-1} &= z_{d-2} \cdot x. \end{aligned}$$

У подальших розділах неодноразово виникатиме задача

Піднесення до степеня за модулем n
Задано: $x \in \mathbb{Z}_n$, $d \in \mathbb{N}$.
Обчислити: $x^d \bmod n$.

Завжди можна вважати, що $d < n$. Інакше можна скористатися теоремою Ойлера, щоб понизити показник. Якщо ми використаємо для розв'язання цієї задачі наведену вище прямолінійну програму (з множенням в кільці $\mathbb{Z}_n$), то отримаємо експоненційний алгоритм. Справді, коли d не набагато менше, ніж n, то довжина входу має порядок $\log n$, а кількість операцій множення має порядок n.

На щастя, є значно ощадніший алгоритм, який був відомий ще до нашої ери в Індії. Ми називатимемо його *бінарним методом*. Опишемо цей метод у вигляді прямолінійної програми для обчислення функції $f(x) = x^d$.

Подамо показник d у двійковій системі числення: $d = (d_l \dots d_1 d_0)_2$, де $d_i \in \{0,1\}$ і $d = \sum_{i=0}^{l} d_i 2^i$. Для спрощення запису дозволимо одній команді програми виконувати до двох множень і покладемо $z_0 = 1$. Тоді i-та команда задається так:

$$z_i = \begin{cases} z_{i-1} \cdot z_{i-1}, & \text{якщо } d_{l+1-i} = 0, \\ z_{i-1} \cdot z_{i-1} \cdot x, & \text{якщо } d_{l+1-i} = 1. \end{cases}$$

Всього команд $l+1$. Результатом виконання останньої є

$$\begin{aligned} (\dots((x^{d_l})^2 x^{d_{l-1}})^2 x^{d_{l-2}} \dots)^2 x^{d_0} &= \\ x^{d_l 2^l + d_{l-1} 2^{l-1} + d_{l-2} 2^{l-2} + \dots + 2^0 d_0} &= x^d. \end{aligned}$$

Всього витрачається $l + \sum_{i=0}^{l-1} d_i \le 2l \le 2\log_2 d$ множень (множення першої команди не є необхідним — вона включена для однорідності). Наведену прямолінійну програму можна легко конвертувати в алгоритм у сенсі пункту 1.2, чи в програму на будь-якій мові програмування, що розв'язує задачу піднесення до степеня за модулем n.

2.3. Функціональні схеми. Будь-яку функцію $F : \{0,1\}^m \to \{0,1\}^k$ можна трактувати як функцію $F : \mathbb{Z}_2{}^m \to \mathbb{Z}_2{}^k$ і розглядати можливість її обчислення прямолінійною програмою. Зауважимо, що стандартні логічні зв'язки можна виразити через арифметичні операції в $\mathbb{Z}_2$: кон'юнкцію булевих змінних x_1 і x_2 як їх добуток $x_1 x_2$, диз'юнкцію як $x_1 \oplus x_2 \oplus x_1 x_2$, а заперечення змінної x як операцію додавання одиниці $x \oplus 1$. Оскільки кожну булеву функцію $f : \{0,1\}^m \to \{0,1\}$ можна виразити булевою формулою у диз'юнктивній нормальній формі, приходимо до висновку, що кожна функція $F : \{0,1\}^m \to \{0,1\}^k$ обчислюється певною прямолінійною програмою. Складність обчислення функції F позначимо через $C(F)$. Таким чином, $C(F)$ позначає найменшу можливу довжину прямолінійної програми над $\mathbb{Z}_2$ для обчислення функції F.

Прямолінійні програми над $\mathbb{Z}_2$ довжини l частіше називають *(функціональними) схемами розміру l у базі* $\{\wedge, \oplus, 1\}$. Як зазначалось, кон'юнкція $\wedge$ збігається із множенням в $\mathbb{Z}_2$. Логічна зв'язка $\oplus$, що відповідає додаванню в $\mathbb{Z}_2$, часом позначається як XOR, від eXclusive OR — "виключне або".

Зазначимо принципову відмінність між схемами та алгоритмами у сенсі пункту 1.2. Перші обчислюють скінченні функції вигляду $f : \{0,1\}^m \to \{0,1\}^k$, в той час як другі — нескінченні функції вигляду $f : \{0,1\}^* \to \{0,1\}^*$ (як зазначалось в пункті 1.2, вибір алфавіту не відіграє принципової ролі, тож ми задля зручності надаємо перевагу двійковому). Однак між двома моделями існує глибокий зв'язок. Наступна теорема справедлива для будь-якої природної формалізації поняття алгоритму.

ТЕОРЕМА 2.2. *Нехай функція $f : \{0,1\}^* \to \{0,1\}^*$ обчислюється деяким поліноміальним алгоритмом. Позначимо через $f_n : \{0,1\}^n \to \{0,1\}^*$ звуження f на множину $\{0,1\}^n$. Тоді $C(f_n) \le cn^c$ для деякої константи $c > 0$.*

Підкреслимо, що обернене твердження в загальному випадку хибне. Пояснити стан речей можна так. Якщо ми маємо послідовність схем S_n для обчислення функцій f_n, то напрошується такий алгоритм обчислення функції f на довільному вході w довжини n:

1) взяти схему S_n;
2) промоделювати її обчислення на вході w.

Так от, з пунктом 2 немає проблем, але для втілення пункту 1 повинен бути *ефективний* спосіб породження схеми S_n за словом довжини n. Послідовність схем $\{S_n\}_{n \in \mathbb{N}}$ з такою властивістю називають *однорідною*. Очевидно, що однорідною є далеко не кожна послідовність схем[1]. В цьому контексті, алгоритми у сенсі пункту 1.2 називають *однорідною моделлю обчислень*, а прямолінійні програми та схеми — *неоднорідною*.

На завершення звернемо увагу ще на один нюанс. Прямолінійні програми над $\mathbb{Z}$ виконують множення чи додавання як "внутрішню" операцію, на що йде один крок. Мультиплікативна складність є більш реалістичною мірою складності, оскільки на практиці множення займає більше обчислювальних ресурсів, ніж додавання. Кажуть, що операція множення є *дорожчою* операцією, або навпаки, додавання є *дешевшою* операцією.

[1] Для читача, знайомого з теорією множин, зазначимо, що однорідних послідовностей є зліченна кількість, а неоднорідних — незліченна.

Функціональні схеми, тобто прямолінійні програми над $\mathbb{Z}_2$, дозволяють розглядати внутрішню структуру операцій множення та додавання у двійковій системі числення. В контексті виконання арифметичних операцій команди прямолінійної програми над $\mathbb{Z}_2$ часто називають *бітовими операціями*. Стандартні алгоритми додавання та множення двох n-розрядних двійкових чисел затрачають, відповідно, $O(n)$ і $O(n^2)$ бітових операцій. Ділення (з остачею) має такий же порядок складності, як і множення [7, розділ 8.2]. Шонгаге і Штрассен запропонували алгоритм, який виконує множення коштом $O(n \log n \log\log n)$ бітових операцій [61] (див. також [7, розділ 7] і [32, розділ 4.3.3]). Їх метод використовується в асимптотично швидкому алгоритмі знаходження НСД двох n-розрядних чисел за $O(n \log^2 n \log\log n)$ операцій [7, розділ 8.10].

ВПРАВИ

2.1. Скласти прямолінійну програму для обчислення в будь-якому кільці загального многочлена степеня n:

$$f(x, x_0, x_1, \ldots, x_n) = x_n x^n + x_{n-1} x^{n-1} + \ldots + x_1 x + x_0.$$

a) Реалізувати обчислення безпосередньо за канонічною формою.

b) Реалізувати обчислення за схемою Горнера.

Порівняти складність програми в першому і другому випадках. (Доведення оптимальності схеми Горнера див. в [35] або [7, розділ 12].)

2.2. Довести, що функцію $f : \mathbb{Z}^2 \to \mathbb{Z}$, задану співвідношенням $f(x_1, x_2) =$ НСД (x_1, x_2), не можна обчислити ніякою прямолінійною програмою.

2.3. Використовуючи бінарний метод, написати прямолінійні програми для обчислення функцій a) x^{32}, b) x^{31}, c) x^{21}.

2.4. Позначимо через $M(d)$ мультиплікативну складність функції $f(x) = x^d$. Довести, що

a) $M(d) \geq \log_2 d$,

b) $M(d_1 d_2) \leq M(d_1) + M(d_2)$.

Див. [32, розділ 4.6.3] за подальшою інформацією про $M(d)$.

2.5. Задачу множення двох комплексних чисел $x_1 + x_2 i$ та $y_1 + y_2 i$ можна трактувати як задачу обчислення двох функцій з $\mathbb{R}^4$ в $\mathbb{R}$

$$f_1(x_1, x_2, y_1, y_2) = x_1 y_1 - x_2 y_2, \quad f_2(x_1, x_2, y_1, y_2) = x_1 y_2 + x_2 y_1.$$

Як видно, мультиплікативна складність цієї задачі не перевищує 4. Скласти для цієї задачі прямолінійну програму з мультиплікативною складністю 3. (Доведення оптимальності такої програми див. в [35] або [7, розділ 12].)

2.6. (Ф. Штрассен [63]) Задачу множення двох матриць розміру 2×2 над кільцем R можна поставити як задачу обчислення чотирьох функцій

$f_{11}, f_{12}, f_{21}, f_{22}$ з R^8 в R кожна, де $f_{ij}(x_{11}, x_{12}, x_{21}, x_{22}, y_{11}, y_{12}, y_{21}, y_{22}) = \sum_{k=1}^{2} x_{ik} y_{kj}$. Стандартний алгоритм множення матриць показує, що мультиплікативна складність цієї задачі не перевищує 8.

a) Придумати прямолінійну програму для множення матриць розміру 2×2 із мультиплікативною складністю 7 (якщо не вдасться, див. ключ до вправ).

b) Позначимо через $M(n)$ мультиплікативну складність множення двох матриць розміру $n \times n$. Довести, що $M(n_1 n_2) \leq M(n_1) M(n_2)$.

c) Традиційний алгоритм множення матриць дає оцінку $M(n) \leq n^3$. Поліпшити цю оцінку до $M(n) \leq 7n^{\log_2 7}$.

2.7. Для натурального числа x позначимо через $C(x)$ складність обчислення прямолінійними програмами функції $f : \mathbb{Z}_2 \to \mathbb{Z}$ тотожньо рівної x.

a) Довести, що $\log_2 \log_2 x \leq C(x) \leq 2 \log_2 x$.

b) Послідовність Фібоначчі задається рекурентним співвідношенням $f_1 = f_2 = 1$, $f_n = f_{n-1} + f_{n-2}$. Звідси безпосередньо випливає, що $C(f_n) < n$. Показати, що ця оцінка відрізняється від оцінки $C(f_n) \leq 2 \log_2(f_n)$ з попереднього пункту в константу разів.

c) (за [62]) Довести оцінку $C(f_n) \leq 24 \log_2 n$.

2.8. Виразити кожен елемент бази $\{\wedge, \oplus, 1\}$ через елементи бази $\{\vee, \wedge, \neg\}$ з кон'юнкції, диз'юнкції і заперечення, та навпаки.

2.9. Довести, що кожна функція $f : \mathbb{Z}_2^m \to \mathbb{Z}_2$ обчислюється поліномом із $\mathbb{Z}_2[x_1, \ldots, x_m]$, причому єдиним, якщо домовитись, що поліном задається канонічною формою, яка не містить степенів x_i^k для $k > 1$, що в $\mathbb{Z}_2$ рівні просто x_i.

2.10. a) Скласти схему в базі $\{\wedge, \oplus\}$ для обчислення функції $f : \{0,1\}^6 \to \{0,1\}$, яка набуває значення 0 на всіх паліндромах і лише на них. Паліндромом називаємо слово $x_1 x_2 x_3 x_4 x_5 x_6$ таке, що $x_1 x_2 x_3 = x_4 x_5 x_6$.

b) Скласти схему в базі $\{\vee, \wedge, \oplus, 1\}$ для обчислення функції $f : \{0,1\}^6 \to \{0,1\}$, яка набуває на вході $x_1 x_2 x_3 x_4 x_5 x_6$ значення 1 тоді і тільки тоді, коли $(x_1 x_2 x_3)_2 < (x_4 x_5 x_6)_2$.

2.11. a) Скласти схему в базі $\{\vee, \wedge, \oplus\}$ для додавання двох дворозрядних двійкових чисел $(x_1 x_0)_2$ і $(y_1 y_0)_2$. Більш формально: позначимо суму через $(u_2 u_1 u_0)_2$. Слід скласти схему для обчислення трьох функцій u_2, u_1, u_0 від чотирьох змінних x_1, x_0, y_1, y_0 кожна.

b) Скласти схему в базі $\{\vee, \wedge, \oplus, 1\}$ для множення таких двох чисел.

ЛІТЕРАТУРА

З алгебраїчної теорії складності можна порадити підручники [35, 39], та огляди [4, 17, 146]. Цьому предметові присвячені розділи в [32, 7] і цикл задач в [31].

§ 3. Рандомізація

3.1. Ймовірнісні алгоритми. Алгоритми, що розглядались досі, називають *детермінованими*. Більші обчислювальні можливості мають *ймовірнісні* алгоритми. Окрім входу w, ймовірнісний алгоритм отримує випадкову двійкову послідовність $r \in \{0,1\}^l$, далі працює як звичайний детермінований алгоритм і результат роботи u подає на вихід. Довжина випадкової послідовності l залежить від довжини входу.

Слід підкреслити, що вихід $u = u(w,r)$ ймовірнісного алгоритму залежить не лише від входу, а й від випадкової послідовності. Випадкова послідовність вважається рівномірно розподіленою на $\{0,1\}^l$, тобто кожне r вибирається з ймовірністю 2^{-l}. Ймовірнісний алгоритм розв'язує масову задачу *з ймовірністю помилки* ϵ, якщо отримавши на вхід індивідуальну задачу w, він подає на вихід її правильний розв'язок з імовірністю не менш ніж $1-\epsilon$. Іншими словами, вихід алгоритму $u = u(w,r)$ є розв'язком індивідуальної задачі w для всіх, окрім щонайбільше $\epsilon 2^l$ випадкових послідовностей r.

Ймовірнісні алгоритми ще називають *алгоритмами Монте Карло*. *Лас Вегас алгоритми* є підкласом алгоритмів Монте Карло. Лас Вегас алгоритм розв'язує задачу з ймовірністю невдачі $\epsilon \in (0,1)$, якщо на кожному вході він видає або правильний розв'язок індивідуальної задачі, або повідомлення (на зразок спеціального символу #) про неспроможність такий розв'язок знайти, причому друге має місце з ймовірністю щонайбільше ϵ. Можна вважати, що Лас Вегас алгоритм взагалі не помиляється, лише часом утримується від подання на вихід розв'язку.

Час роботи ймовірнісного алгоритму на вході w означується як максимальна по r кількість кроків, які алгоритм робить на цьому вході з використанням випадкової послідовності r.[2] Таким чином поняття поліноміального часу, яке було введене для детермінованих алгоритмів, переноситься й на клас ймовірнісних алгоритмів.

Окремо зупинимось на ймовірнісних алгоритмах для задач розпізнавання. Ймовірнісний (Монте Карло) алгоритм A розпізнає мову L з *однобічною помилкою* ϵ, де $0 < \epsilon < 1$, якщо дотримані такі дві умови:
1) якщо $w \in L$, то A приймає вхід w з імовірністю 1;

[2] Іноді час роботи ймовірнісного алгоритму означують як середню, а не максимальну кількість кроків.

2) якщо $w \notin L$, то A відхиляє вхід w з імовірністю принаймні $1-\epsilon$.

Нехай ймовірнісний алгоритм A розпізнає мову L з однобічною помилкою ϵ, використовуючи на вході довжини n випадкову послідовність довжини $l = l(n)$. В цьому випадку є ефективний спосіб зменшення помилки. Розглянемо ймовірнісний алгоритм A^t, який на вході w працює наступним чином:

- використовує випадкову послідовність довжини lt, де $l = l(|w|)$, розбивши її на t частин $r_1, \ldots, r_t$ довжини l кожна;
- моделює роботу алгоритму A на вході w з використанням кожної із випадкових послідовностей $r_1, \ldots, r_t$, і отримує t результатів моделювання $u_1, \ldots, u_t$, де $u_i \in \{0,1\}$;
- подає на вихід 1, якщо всі u_i дорівнюють 1, і 0, якщо серед них є хоча б один 0.

Перевіримо, що алгоритм A^t розпізнає ту ж мову L і оцінимо ймовірність помилки. Якщо $w \in L$, то алгоритм A видає 1 незалежно від вибору випадкової послідовності. Тому $u_i = 1$ для всіх $i \le t$, і у цьому випадку A^t приймає w з ймовірністю 1. Якщо $w \notin L$, то $u_i = 1$ з ймовірністю щонайбільше ϵ, для кожного i. Оскільки $r_1, \ldots, r_t$ є випадковими і незалежними між собою, то рівність $u_i = 1$ виконується для всіх $i \le t$ з ймовірністю не більшою ніж ϵ^t. Приходимо до висновку, що A^t приймає w не з L із ймовірністю щонайбільше ϵ^t і, таким чином, розпізнає L з однобічною помилкою ϵ^t.

Важливе зауваження полягає в тому, що якщо алгоритм A поліноміальний, то таким є і алгоритм A^t, причому не лише для t константи, а й для будь-якого $t(n)$ — поліному від довжини входу. Поклавши $t(n) = cn$ для відповідної константи c, бачимо, що якщо множина розпізнається поліноміальним ймовірнісним алгоритмом із однобічною помилкою ϵ, для константи ϵ між 0 і 1, то ця ж множина розпізнається деяким поліноміальним алгоритмом із однобічною помилкою 2^{-n}. Ймовірність такого порядку називають *експоненційно низькою*. Таким чином, експоненційне зниження помилки (до ϵ^t) досягається ціною всього лише лінійного збільшення часу роботи (у t разів).

3.2. Випадковий вибір. Генерування послідовності випадкових бітів на практиці є не такою простою справою, як може видатись на перший погляд, адже реальна обчислювальна машина не має ніякого механізму для "підкидання монетки". Ми повернемось до цього питання в розділі VI, а до того моменту будемо вважати, що ймовірнісний алгоритм здатен отримати як завгодно довгу послідовність випадкових

бітів, причому на отримання одного біту йде один крок роботи алгоритму.

У подальшому викладі конкретних ймовірнісних алгоритмів ми вживатимемо приписи на кшталт *"вибрати випадковий елемент x із скінченної множини X"*. Це означає, що елемент x має бути сконструйованим певним чином із випадкової послідовності r, причому кожен з елементів множини X мусить отримуватися *з однаковою ймовірністю*. Спосіб такого конструювання найчастіше буде доволі очевидним. У цьому пункті ми заздалегідь розберемо кілька простих випадків, які далі зустрічатимуться без детальних пояснень.

Найпростіше влаштувати випадковий вибір елемента із множини $\mathbb{Z}_n$ для $n = 2^l$, або із $\mathbb{Z}_p^*$ для простого $p = 2^l + 1$.[3] Для цього випадкова двійкова послідовність r довжини l просто розглядається як двійковий запис елемента такої множини.

Щоб отримати випадковий елемент із $\mathbb{Z}_n$ для довільного n, можна вибрати випадковий елемент x із $\mathbb{Z}_{2^l}$, де $l = \lceil \log_2 n \rceil$, і якщо $x < n$, то використовувати цей елемент, якщо ж $x \geq n$, то вибір слід повторити ще раз. Внаслідок цієї процедури буде отримано $x \in \mathbb{Z}_n$ з ймовірністю $\alpha > 1/2$. Зрозуміло, що в разі невдачі повторювати вибір доведеться не надто довго. Справді, ймовірність того, що x попаде в $\mathbb{Z}_n$ лише за i-тим разом, дорівнює $(1-\alpha)^{i-1}\alpha$, а математичне сподівання кількості повторень процедури до першого успішного вибору дорівнює

$$1\alpha + 2(1-\alpha)\alpha + 3(1-\alpha)^2\alpha + 4(1-\alpha)^3\alpha + \cdots = \frac{1}{\alpha} < 2.$$

Таким же чином проводиться вибір випадкового елемента із $\mathbb{Z}_p^*$ для довільного простого p. Щоб вибрати випадковий елемент з множини $\mathbb{Z}_{pq}^*$, де p і q прості, можна скористатися наслідком II.2.13 з Китайської теореми про остачі. Згідно із ним, досить вибрати x_1 і x_2 — випадкові елементи множин $\mathbb{Z}_p^*$ і $\mathbb{Z}_q^*$, і обчислити $x \in \mathbb{Z}_{pq}^*$, для якого $x_1 = x \bmod p$ і $x_2 = x \bmod q$. Такий елемент x рівномірно розподілений на $\mathbb{Z}_{pq}^*$.

Менш вишуканий, але простіший і ефективніший для великих p і q спосіб породження випадкового елемента з $\mathbb{Z}_{pq}^*$ є таким. Вибираємо випадковий елемент x з $\mathbb{Z}_{pq}$ і перевіряємо, чи він ділиться на якесь із чисел p і q. Якщо ні, то $x \in \mathbb{Z}_{pq}^*$, що нам і потрібно, якщо ж ділиться, то процедуру вибору повторюємо ще раз. Ймовірність того, що вибір буде успішно здійснено за один раз, дорівнює $\phi(pq)/pq = (1-1/p)(1-1/q)$.

[3] Прості числа такого вигляду називаються *простими Ферма*.

Цей спосіб добрий також тим, що він придатний для породження випадкового елементу множини $\mathbb{Z}_n^*$ для довільного n. А саме, вибирається випадковий елемент x з $\mathbb{Z}_n$ і обчислюється НСД(x, n). Якщо останній дорівнює 1, то вибір здійснено, а інакше вибір слід повторити. Ймовірність того, що вибір буде успішно зроблено за один раз, дорівнює $\phi(n)/n$. Оцінимо, наскільки малою є ймовірність того, що елемент із $\mathbb{Z}_n^*$ все ще не буде вибрано упродовж $6k \ln\ln n$ повторень процедури вибору. Ця ймовірність дорівнює $(1-\phi(n)/n)^{6k\ln\ln n}$. За твердженням II.2.15, $\phi(n)/n > 1/(6\ln\ln n)$ при великих n. Отже, ймовірність, яку ми оцінюємо, обмежена зверху величиною $(1-1/(6\ln\ln n))^{6k\ln\ln n}$, яка, у свою чергу, не перевищує e^{-k} за нерівністю B.4. Підсумовуючи, бачимо, що з високою ймовірністю (наскільки високою, залежить від нашого вибору константи k) випадковий елемент x з $\mathbb{Z}_n^*$ можна вибрати в результаті не більше, як $6k \ln\ln n$ незалежних виборів випадкового елемента x з $\mathbb{Z}_n$ і перевірки умови НСД$(x, n) = 1$.

У цьому контексті доречно згадати класичний результат Діріхле. Нехай x — випадковий елемент із множини $\mathbb{Z}_n$, де n також вибрано випадковим чином в діапазоні від 1 до N. Тоді ймовірність того, що НСД$(x, n) = 1$ при зростаючому N прямує до $6/\pi^2 \approx 0.6$ (див. вправу 3.9).

ВПРАВИ

3.1. Припустимо, що ймовірнісний алгоритм A розв'язує задачу П з ймовірністю помилки $1/2 - \epsilon$, де $0 < \epsilon < 1/2$ (помилка не обов'язково однобічна!). Припустимо також, що кожна індивідуальна задача масової задачі П має єдиний розв'язок — такими є, зокрема, всі задачі обчислення і розпізнавання. Розглянемо такий спосіб зменшення ймовірності помилки. Нехай алгоритм A^t викликає алгоритм A як підпроцедуру непарну кількість разів t, щоразу із незалежним вибором випадкової послідовності бітів. На вихід алгоритм A^t подає результат, що трапився більш ніж у $t/2$ випадках (якщо ж такого не існує, то видає спеціальний символ #). Довести, що алгоритм A^t розв'язує задачу П з ймовірністю помилки $e^{-\epsilon^2 t}$.

3.2. Довести, що якщо мова $L \subseteq \mathcal{A}^*$ розпізнається поліноміальним Лас Вегас алгоритмом з ймовірністю невдачі ϵ, то доповнення до неї $\mathcal{A}^* \setminus L$ також розпізнається деяким поліноміальним Лас Вегас алгоритмом з такою ж ймовірністю невдачі.

3.3. Довести, що мова $L \subseteq \mathcal{A}^*$ розпізнається поліноміальним Лас Вегас алгоритмом з імовірністю невдачі ϵ тоді і лише тоді, коли одночасно і L, і її доповнення $\mathcal{A}^* \setminus L$ розпізнаються поліноміальними Монте Карло алгоритмами з однобічною помилкою ϵ.

3.4. Розглянемо різновид ймовірнісного алгоритму, який має доступ до потенційно необмеженої послідовності випадкових бітів $r \in \{0,1\}^{\mathbf{N}}$ (в ході обчислення може використовуватись лише початковий відрізок r, але як завгодно довгий). За означенням такий алгоритм розпізнає мову L з ймовірністю 1. Це означає, що на кожному вході w для майже всіх r (за винятком множини міри 0 відносно міри Лебега λ) алгоритм зупиняється через скінченну кількість кроків, приймаючи $w \in L$ і відхиляючи $w \notin L$. Позначимо через $t(w,r)$ кількість кроків алгоритму на вході w з випадковою послідовністю r. Час роботи алгоритму на вході w означимо як інтеграл Лебега $\int t(w,r)\,d\lambda(r)$. Довести, що мова L розпізнається у такій моделі за поліноміальний час тоді і тільки тоді, коли вона розпізнається деяким поліноміальним Лас Вегас алгоритмом.

3.5. *Цілий вираз* над змінними $x_1,\dots,x_m$ означується наступним чином:

— десятковий запис цілого числа є цілим виразом;

— будь-яка із змінних $x_1,\dots,x_m$ є цілим виразом;

— якщо P і Q — цілі вирази, то $(P\cdot Q)$, $(P+Q)$, $(P-Q)$ є цілими виразами.

Наприклад, $((x_1-x_2)\cdot(x_2+x_3)+x_2\cdot(x_1+x_3))$ та $(x_1\cdot(x_1+x_3)-(x_1-x_2)\cdot(x_1-x_2))$ є цілими виразами. Два цілі вирази назвемо тотожними, якщо вони представляють один і той же поліном з цілими коефіцієнтами від змінних $x_1,\dots,x_m$. Наприклад, два наведені вирази тотожні — обидва представляють поліном, канонічна форма якого $-x_2^2+2x_1x_2+x_1x_3$.

Невідомо, чи існує поліноміальний детермінований алгоритм для розпізнавання тотожності двох заданих цілих виразів. Зауважимо, що в загальному випадку цілий вираз не можна звести до канонічної форми за поліноміальний час з тої причини, що його канонічна форма може мати експоненційний розмір (як, наприклад, для виразу $(x_1+1)(x_2+1)\dots(x_m+1)$). Метою вправи є розробити для цієї задачі поліноміальний ймовірнісний алгоритм з однобічною помилкою.

a) Степенем монома $x_1^{i_1}x_2^{i_2}\dots x_m^{i_m}$ є сума показників $i_1+i_2+\dots+i_m$. Степенем полінома називається найбільший степінь монома у його канонічній формі. Довести, що степінь полінома, представленого цілим виразом, не перевищує довжини запису останнього (тобто загальної кількості символів $(,),+,-,\cdot,x_1,\dots,x_m$).

b) (J. T. Schwartz, R. E. Zippel) Нехай F — поле і H — множина із h елементів цього поля. Нехай $p(x_1,\dots,x_m)$ — поліном в $F[x_1,\dots,x_m]$ степеня d. Нулем полінома p називається такий набір $(\alpha_1,\dots,\alpha_m)\in F^m$, що $p(\alpha_1,\dots,\alpha_m)=0$. Довести, що множина H^m містить щонайбільше dh^{m-1} нулів полінома p.

c) Оцінити час роботи і ймовірність помилки наступного алгоритму розпізнавання тотожності цілих виразів. Нехай на вхід подаються цілі вирази P і Q від m змінних, кожен довжини щонайбільше n. Випадково і незалежно вибираються m цілих чисел $\alpha_1,\dots,\alpha_m$ в межах від 0 до n^2 кожне. Вхід приймається, якщо $P(\alpha_1,\dots,\alpha_m)=Q(\alpha_1,\dots,\alpha_m)$ і відхиляється інакше.

3.6. Припустимо, що $X\subseteq Y$, де Y — скінченна множина. Розглянемо такий ймовірнісний експеримент із нескінченної кількості кроків. На i-му кроці з множини Y вибирається випадковий елемент y_i, причому вибір здійснюється рівноймовірно і незалежно від попередніх кроків. Позначимо через x перший елемент в послідовності $y_1,y_2,y_3,\dots$, який належить множині X, а через k — номер кроку, для якого вперше $y_k=x\in X$.

a) Довести, що випадкова величина x рівномірно розподілена на множині X.

b) Чому дорівнює математичне сподівання випадкової величини k?

Нагадування. При розв'язуванні наступних задач слід пам'ятати, що за домовленістю випадковий елемент множини набуває кожне значення з неї з однаковою ймовірністю.

3.7. Нехай m і n — натуральні числа, і x є випадковим елементом множини $\mathbb{Z}_n$. Довести, що $x \bmod m$ є випадковим елементом множини $\mathbb{Z}_m$ тоді і тільки тоді, коли n ділиться на m.

3.8. a) Нехай a — довільний *фіксований* елемент групи G, а r — *випадковий* елемент цієї групи. Довести, що їх добуток ar є випадковим елементом групи G.

b) Довести, що добуток випадкових елементів групи є випадковим елементом цієї групи.

3.9. Нехай натуральні числа x і y вибираються випадково і незалежно в діапазоні від 1 до N. Довести, що ймовірність події НСД$(x,y)=1$ дорівнює

$$\frac{2(1+\phi(2)+\phi(3)+\dots+\phi(N))-1}{N^2}.$$

Зауваження. При $N\to\infty$ останній вираз прямує до $1/\zeta(2)$, де $\zeta(z)$ — *дзета функція Рімана*, яка буде означена у пункті IV.2.6. Доведення цього факту можна знайти у [13, розділ II, вправа 21]. Справедлива також рівність $\zeta(2)=\pi^2/6$ (див. напр. [15]).

§ 4. Порівняння складності задач

4.1. Оракульна модель. Опишемо поняття *оракульного алгоритму*. Це абстрактна обчислювальна модель, яка не описує ніяких реальних обчислень, а потрібна нам для порівняння обчислювальної складності задач.

Під *оракулом* ми будемо розуміти будь-яку функцію $\mathcal{O}:\mathcal{A}^*\to\mathcal{B}^*$, де $\mathcal{A}$ і $\mathcal{B}$ — скінченні алфавіти. Оракульний алгоритм є алгоритмом у

звичному розумінні, але наділений додатковою елементарною операцією *звернення до оракула*. Ця операція виконується зразу після того, як алгоритм в результаті своєї попередньої роботи надрукував деяке слово $z \in \mathcal{A}^*$, що називається *запитом до оракула*. Внаслідок звернення до оракула $\mathcal{O}$ алгоритм отримує слово $\mathcal{O}(z)$, *відповідь оракула*, яке використовується у подальшій роботі. Оракульний алгоритм A із *доступом до оракула* $\mathcal{O}$ позначатимемо через $A^{\mathcal{O}}$. Зауважимо, що робота оракульного алгоритму може суттєво залежати від того, до якого конкретно оракула він має доступ.

Як приклад наведемо щонайпростіший алгоритм A з доступом до оракула $\mathcal{O} : \{0,1\}^* \to \{0,1\}^*$.

Алгоритм A

- отримує на вхід слово $w \in \{0,1\}^*$;
- звертається до оракула із запитом w;
- подає на вихід $\mathcal{O}(w)$.

Зрозуміло, що $A^{\mathcal{O}}$ обчислює функцію $\mathcal{O} : \{0,1\}^* \to \{0,1\}^*$. Цей приклад демонструє наскільки широкими (і нереалістичними) є можливості оракульних алгоритмів порівняно із звичайними — оракульний алгоритм може обчислити навіть таку функцію, яку не можна обчислити ніяким звичайним алгоритмом, варто лише взяти достатньо "потужний" оракул.

Поняття часу роботи поширюється на оракульні алгоритми згідно із домовленістю, що звернення до оракула займає лише один крок. Але окремо слід враховувати час, який затрачається на друкування запиту та зчитування відповіді оракула. Оракульний алгоритм називається поліноміальним, якщо він працює час, обмежений деяким поліномом від довжини входу, причому це обмеження виконується для кожного оракула. Звернення до оракула $\mathcal{O}$ природно інтерпретувати у програмістських термінах як виклик програмою підпроцедури, що обчислює функцію $\mathcal{O}$ на заданому аргументі, причому час роботи процедури не враховується при оцінці загального часу програми.

Оракул $\mathcal{O}$, що є характеристичною функцією деякої множини $L \subseteq \mathcal{A}^*$, будемо утотожнювати із самою множиною і писати A^L замість $A^{\mathcal{O}}$.

4.2. Звідності задач. Оракульний алгоритм A *зводить* задачу Π_1 до задачі Π_2, якщо для будь-якого оракула $\mathcal{O}$, який кожну індивідуальну задачу w масової задачі Π_2 відображає у якийсь із її розв'язків, алгоритм $A^{\mathcal{O}}$ розв'язує задачу Π_1. Зазначимо, що якщо Π_2 є задачею обчислення або розпізнавання, то означення спрощується, адже в цих

випадках оракул $\mathcal{O}$, про який йдеться, визначений однозначно. Таким чином, оракульний алгоритм A зводить задачу Π до задачі обчислення функції f, якщо алгоритм A^f розв'язує задачу Π. Також, оракульний алгоритм A зводить задачу Π до задачі розпізнавання множини L, якщо алгоритм A^L розв'язує задачу Π.

Якщо алгоритм A зводить Π_1 до Π_2, то кажуть, що він є *зведенням* першої задачі до другої. Якщо існує зведення A задачі Π_1 до задачі Π_2, яке є поліноміальним алгоритмом, то кажуть, що Π_1 *поліноміально зводиться до* Π_2, а A називають *поліноміальним зведенням*.

Приклад 4.1. Нехай Π_1 — задача знаходження нетривіального дільника заданого натурального числа, а Π_2 — задача розпізнавання такої множини $L \subset \mathbb{N}^2$:

$$L = \{\, (y,x) : \text{ існує } z \text{ таке, що } 1 < z \le y,\ z \ne x,\ z \mid x \,\}.$$

Π_1 поліноміально зводиться до Π_2 за допомогою алгоритму *бінарного пошуку*.

Вхід: $x \in \mathbb{N}$.

- Якщо $(x,x) \notin L$ (звернення до оракула!), то видати повідомлення, що x просте і припинити роботу.
- Інакше покласти $a_0 = 1$ і $b_0 = x$. Подальша робота алгоритму розбивається на кроки. i-ий крок полягає в наступному.
- Якщо $(\lceil \frac{a_{i-1}+b_{i-1}}{2} \rceil, x) \in L$ (чергове звернення до оракула), то покласти $a_i = a_{i-1}$, $b_i = \lceil \frac{a_{i-1}+b_{i-1}}{2} \rceil$, а інакше $a_i = \lceil \frac{a_{i-1}+b_{i-1}}{2} \rceil$, $b_i = b_{i-1}$.
- Якщо $b_i - a_i = 1$, то подати на вихід b_i і припинити роботу, а інакше перейти до виконання наступного $(i+1)$-го кроку.

Неважко зрозуміти, що описаний алгоритм або визначає, що вхід x є простим числом, або знаходить його найменший простий дільник. Оцінимо час роботи цього зведення. Як легко зауважити, $b_i - a_i < (b_{i-1} - a_{i-1})/2 + 1$. Звідси випливає, що $b_i - a_i < (b_0 - a_0)/2^i + 2$. Отже, на вході x алгоритм робить не більше $\log_2 x + 2$ кроків і справді є поліноміальним.

Якщо Π_1 поліноміально зводиться до Π_2, то говоримо, що задача Π_1 *не важча від задачі* Π_2, а Π_2 *не легша, ніж* Π_1.

Поняття оракульного алгоритму без труднощів поширюється і на ймовірнісні алгоритми. Таким чином отримуємо поняття ймовірнісного поліноміального зведення однієї задачі до іншої.

ТЕОРЕМА 4.2. *Нехай* Π_1, Π_2 *і* Π_3 *є алгоритмічними задачами.*

1) *Якщо* Π_1 *поліноміально зводиться до* Π_2, *а* Π_2 *до* Π_3, *то* Π_1 *поліноміально зводиться до* Π_3.
2) *Якщо задача* Π_1 *поліноміально зводиться до задачі* Π_2, *яка розв'язується за поліноміальний час, то* Π_1 *теж розв'язується за поліноміальний час.*
3) *Якщо існує ймовірнісне поліноміальне зведення* Π_1 *до* Π_2 *з деякою ймовірністю помилки, і задача* Π_2 *розв'язується за поліноміальний час, то* Π_1 *розв'язується деяким поліноміальним ймовірнісним алгоритмом з такою ж ймовірністю помилки.* ▪

Доведення теореми залишається читачеві у якості нескладної вправи.

Якщо задача Π_1 поліноміально зводиться до задачі Π_2, а Π_2 поліноміально зводиться до Π_1, то такі задачі називаються *поліноміально еквівалентними.*

ПРИКЛАД 4.3. Нехай Π_1 — задача знаходження нетривіального дільника заданого натурального числа, а Π_2 — задача знаходження канонічного розкладу числа на прості співмножники. Ці дві задачі поліноміально еквівалентні. Зведення Π_1 до Π_2 очевидне. Продемонструємо обернене зведення, тобто опишемо алгоритм A, який безкоштовно отримує розв'язки задачі Π_1 і знаходить канонічний розклад заданого числа на множники. Отримавши на вхід $x \in \mathbb{N}$, алгоритм A просить дати йому нетривіальний дільник цього числа. Якщо A отримує відповідь, що такого не існує, то подає на вихід розклад, який складається з самого числа x. Якщо ж A отримує нетривіальний дільник y, то далі рекурсивно викликає самого себе на входах y і x/y, і отримавши розклади на множники цих чисел, об'єднує їх у розклад числа x.

Поліноміальна звідність визначає відношення часткового порядку на множині класів еквівалентності алгоритмічних задач.

ВПРАВИ

4.1. Прослідкувати роботу зведення із прикладу 4.1 на вході 35.
4.2. Довести теорему 4.2.
4.3. Довести, що звідність із прикладу 4.3 є поліноміальною.

Розділ IV.

Складність арифметичних задач

У цьому розділі ми вивчаємо складність алгоритмічних задач теорії чисел, на яких грунтується сучасна криптографія.

Насамперед варто зафіксувати певну домовленість стосовно позначень. Коли в теорії чисел йдеться про параметр, що є натуральним числом, його часто позначають через n. Коли в теорії складності йдеться про час роботи алгоритму, n є традиційним позначенням для довжини входу. Обидві традиції не можуть бути дотриманими одночасно, коли йдеться про алгоритм, який отримує на вхід натуральне число. Справді, якщо натуральне число n подається в системі числення за основою k, то його запис має довжину $\lfloor \log_k n \rfloor + 1$. Щоб уникнути двозначності, в цьому розділі ми будемо оцінювати час роботи числового алгоритму як функцію не від *довжини*, а від *величини* входу.

Зауважимо, що алгоритм є поліноміальним тоді і тільки тоді, коли час його роботи на вході n (довжини $\lfloor \log_k n \rfloor + 1!$) обмежений функцією $c(\log_k n)^d$ для деяких констант $c > 0$ і $d > 1$. Алгоритм є експоненційним, якщо на вході n (точніше, на нескінченній послідовності таких входів) час його роботи перевищує cn^d для деяких констант c, $d > 0$.

§ 1. Арифметика II

Перший параграф цього розділу має підготовчий характер. Він містить чергову порцію фактів із теорії чисел, потрібних для побудови та аналізу алгоритмів у наступних параграфах.

1.1. Первісні корені. Нехай p — натуральне число. Ціле число g називається *первісним коренем за модулем* p, якщо лишок $g \bmod p$ є твірним елементом групи $\mathbb{Z}_p^*$.

Для простого p група $\mathbb{Z}_p^*$, як мультиплікативна група скінченного поля, є циклічною. Отже, первісні корені існують для всіх простих модулів. Як легко зрозуміти, g є первісним коренем за простим модулем p, якщо послідовність

$$g^0 \bmod p = 1, g^1 \bmod p, g^2 \bmod p, g^3 \bmod p, \ldots, g^{p-2} \bmod p \qquad (1)$$

містить всі елементи множини $\mathbb{Z}_p^*$. Остання умова очевидно рівносильна тому, що всі члени послідовності (1) попарно різні.

Приклад 1.1. $g = 5$ є первісним коренем за модулем $p = 23$. Послідовність (1) в цьому випадку буде такою: 1, 5, 2, 10, 4, 20, 8, 17, 16, 11, 9, 22, 18, 21, 13, 19, 3, 15, 6, 7, 12, 14. Зауважимо, що черговий член $g^k \bmod p$ не обов'язково обчислювати піднесенням g до k-го степеня. Простіше домножити вже обчислений попередній член $g^{k-1} \bmod p$ на g за модулем p.

Твердження 1.2. *Нехай p позначає просте число, і $p-1 = q_1^{\alpha_1} \ldots q_s^{\alpha_s}$ — канонічний розклад числа $p-1$ на прості співмножники. Тоді*

1) *в $\mathbb{Z}_p^*$ міститься рівно $\phi(p-1)$ первісних коренів за модулем p;*
2) *щоб число g було первісним коренем за модулем p необхідно і досить, щоб для кожного $i \le s$ виконувалась умова*

$$g^{(p-1)/q_i} \not\equiv 1 \,(\mathrm{mod}\, p). \qquad (2)$$

Доведення. Пункт 1 випливає безпосередньо із твердження Б.1.

Доведемо пункт 2. Це досить зробити для $g \in \mathbb{Z}_p^*$. Позначимо через m порядок елемента g в групі $\mathbb{Z}_p^*$. Нагадаємо, що g є первісним коренем за модулем p тоді і тільки тоді, коли $m = p-1$.

Якщо умова (2) порушується хоча б для якогось i, то для такого i маємо $m \le (p-1)/q_i < p-1$.

Навпаки, припустимо, що умова (2) виконується для всіх i. В цьому випадку повинно бути $m = p-1$. Справді, якби $m < p-1$, то за теоремою Лагранжа ми мали б рівність $p-1 = km$ для деякого $k > 1$, і як наслідок $g^{(p-1)/q} \equiv 1 \,(\mathrm{mod}\, p)$ для довільного простого дільника q числа k — суперечність з (2). ∎

Приклад 1.3. Для $p = 29$ маємо $29-1 = 2^2 \cdot 7$. Оскільки $2^{14} \equiv 28 \,(\mathrm{mod}\, 29)$ і $2^4 \equiv 16 \,(\mathrm{mod}\, 29)$, то $g = 2$ є первісним коренем за модулем $p = 29$.

На противагу цьому, $g = 5$ не є первісним коренем за модулем $p = 29$, бо $5^{14} \equiv 1 \,(\mathrm{mod}\, 29)$.

1.2. Квадратичні лишки. Нехай n — натуральне число. Ціле x називається *квадратичним лишком за модулем* n, якщо НСД $(x, n) = 1$ і $x \equiv y^2 \,(\mathrm{mod}\, n)$ для деякого y. В цьому разі y називається *квадратним коренем* з x за модулем n. Якщо НСД $(x, n) = 1$ і x не є квадратичним лишком за модулем n, то x називається *квадратичним нелишком* за цим модулем. Квадратичні лишки за модулем n, які лежать в межах від 1 до $n-1$, називаються *зведеними*. Множину зведених квадратичних лишків будемо позначати через $\mathcal{Q}_n$.

Приклад 1.4. Число 8 є квадратичним лишком за модулем 17. Квадратними коренями з 8 за цим модулем серед натуральних чисел, що не перевищують 17, є 5 і 12. Число 12 є квадратичним нелишком за модулем 17 — воно не має жодного квадратного кореня за цим модулем.

Лема 1.5. *Нехай p — непарне просте число. Тоді наступні умови еквівалентні.*

1) *x є квадратичним лишком за модулем p.*
2) $x^{(p-1)/2} \equiv 1 \,(\mathrm{mod}\, p)$.
3) *Якщо $g \in \mathbb{Z}_p^*$ — первісний корінь за модулем p, то для деякого парного k в $\mathbb{Z}_p^*$ виконується рівність $x = g^k$.*

Доведення. 1) $\Rightarrow$ 2) Використовуємо те, що $x \equiv y^2 \,(\mathrm{mod}\, n)$ для деякого y. Тоді рівність 2 справедлива за малою теоремою Ферма.

2) $\Rightarrow$ 3) З умови 2 випливає, що x не ділиться на p. Оскільки g — первісний корінь за модулем p, то $x = g^k$ для деякого k. Щоб показати, що k парне, припустимо супротивне. Нехай $x = g^{2j+1}$ для деякого j. Використовуючи малу теорему Ферма, отримуємо

$$x^{(p-1)/2} = g^{(p-1)j} g^{(p-1)/2} = g^{(p-1)/2}.$$

За умовою 2 звідси випливає, що $g^{(p-1)/2} = 1$, а це суперечить тому, що g — первісний корінь.

3) $\Rightarrow$ 1) Якщо $x \equiv g^{2j} \,(\mathrm{mod}\, p)$, то g^j є квадратним коренем з x за модулем p. ∎

Для цілого x і непарного простого p *символ Лежандра* означується як

$$\left(\frac{x}{p}\right) = \begin{cases} 1, & \text{якщо } x \text{ є квадратичним лишком за } \bmod p; \\ -1, & \text{якщо } x \text{ є квадратичним нелишком за } \bmod p; \\ 0, & \text{якщо } p \mid x. \end{cases}$$

Критерій Ойлера. *Нехай x — ціле число, а p — непарне просте. Тоді*

$$\left(\frac{x}{p}\right) \equiv x^{(p-1)/2} \pmod{p}.$$

Доведення. У випадку $p \nmid x$ використовуємо еквівалентність умов 1 і 2 з леми 1.5. Залишається показати, що $x^{(p-1)/2} \equiv \pm 1 \pmod{p}$. Останнє випливає з того, що $x^{(p-1)/2}$ є коренем многочлена $X^2 - 1$ в полі $\mathbb{Z}_p^*$, який не може мати більше, ніж два корені 1 і -1. ∎

Твердження 1.6. *Нехай p — непарне просте, а x_1 і x_2 — цілі числа. Символ Лежандра має такі властивості.*

1) *Якщо* $x_1 \equiv x_2 \pmod{p}$, *то* $\left(\frac{x_1}{p}\right) = \left(\frac{x_2}{p}\right)$.

2) *Мультиплікативність:* $\left(\frac{x_1}{p}\right)\left(\frac{x_2}{p}\right) = \left(\frac{x_1 x_2}{p}\right)$.

3) *Якщо* x_2 *не ділиться на* p, *то* $\left(\frac{x_1 x_2^2}{p}\right) = \left(\frac{x_1}{p}\right)$.

Доведення. Пункт 1 випливає з означення символу Лежандра, а пункт 2 — з критерію Ойлера. Пункт 3 є наслідком пункту 2, оскільки $\left(\frac{x_2^2}{p}\right) = 1$ — адже очевидно, що x_2^2 є квадратичним лишком за модулем p. ∎

Нехай $n \geq 3$ — непарне число з розкладом на прості множники $n = p_1^{\alpha_1} \cdot \ldots \cdot p_s^{\alpha_s}$. Нехай x — довільне ціле. Тоді *символ Якобі* $\left(\frac{x}{n}\right)$, що є узагальненням символу Лежандра, означується як

$$\left(\frac{x}{n}\right) = \left(\frac{x}{p_1}\right)^{\alpha_1} \cdot \ldots \cdot \left(\frac{x}{p_s}\right)^{\alpha_s},$$

де множники в правій частині є символами Лежандра.

Твердження 1.7. *Нехай x, x_1, x_2 — цілі числа, а n, n_1, n_2 — непарні не менші від 3 числа. Символ Якобі має такі властивості.*

1) *Якщо* $x_1 \equiv x_2 \pmod{n}$, *то* $\left(\frac{x_1}{n}\right) = \left(\frac{x_2}{n}\right)$.

2) $\left(\frac{x_1}{n}\right)\left(\frac{x_2}{n}\right) = \left(\frac{x_1 x_2}{n}\right)$.

3) *Якщо* x_2 *і* n *взаємно прості, то* $\left(\frac{x_1 x_2^2}{n}\right) = \left(\frac{x_1}{n}\right)$.

4) $\left(\frac{x}{n_1 n_2}\right) = \left(\frac{x}{n_1}\right)\left(\frac{x}{n_2}\right)$.

5) *Якщо x квадратичний лишок за модулем n, то* $\left(\frac{x}{n}\right) = 1$.

Доведення. Властивості 1 і 2 успадковуються від відповідних властивостей символу Лежандра із твердження 1.6. Пункт 4 справедливий за означенням символу Якобі. Пункт 5 випливає з означень символів Якобі та Лежандра і спостереження, що якщо x є квадратичним лишком за модулем n, то x є квадратичним лишком також за модулем p для будь-якого p, що ділить n. Пункт 3 є простим наслідком пунктів 2 і 5. ∎

Слід застерегти, що на відміну від символу Лежандра, для символу Якобі обернене до пункту 5 твердження не має місця. Рівність $\left(\frac{x}{n}\right) = 1$ можлива навіть тоді, коли x є квадратичним нелишком за модулем n, наприклад, число 2 не є квадратичним лишком за жодним із модулів 3, 5, і 15, але $\left(\frac{2}{15}\right) = \left(\frac{2}{3}\right)\left(\frac{2}{5}\right) = (-1)(-1) = 1$.

Наступна теорема є одним із центральних результатів теорії чисел.

Квадратичний закон взаємності Гауса (1796). *Нехай m, $n > 2$ взаємно прості непарні натуральні числа. Тоді*

1) $\left(\frac{m}{n}\right)\left(\frac{n}{m}\right) = (-1)^{(m-1)(n-1)/4}$;

2) $\left(\frac{2}{n}\right) = (-1)^{(n^2-1)/8}$. ∎

Квадратичний закон взаємності вказує шлях швидкого обчислення $\left(\frac{x}{n}\right)$ для заданих x і n.

Алгоритм обчислення символу Якобі. Можна вважати, що $0 \leq x < n$, оскільки $\left(\frac{x}{n}\right) = \left(\frac{x \bmod n}{n}\right)$ на підставі пункту 1 твердження 1.7. Кожен наступний крок алгоритму зводить обчислення $\left(\frac{x}{n}\right)$ до обчислення $\left(\frac{x'}{n'}\right)$, де $x' < x$ і $n' \leq n$.

- Якщо $x = 2^{2j}y$, то за пунктом 3 твердження 1.7 маємо $\left(\frac{x}{n}\right) = \left(\frac{y}{n}\right)$.
- Якщо $x = 2^{2j+1}y$, то за пунктами 3 і 2 твердження 1.7 маємо $\left(\frac{x}{n}\right) = \left(\frac{2y}{n}\right) = \left(\frac{2}{n}\right)\left(\frac{y}{n}\right)$. На підставі пункту 2 квадратичного закону взаємності отримуємо $\left(\frac{x}{n}\right) = \left(\frac{y}{n}\right)(-1)^{(n^2-1)/8}$.
- Якщо x непарне, то за пунктом 1 квадратичного закону взаємності маємо $\left(\frac{x}{n}\right) = \left(\frac{n \bmod x}{x}\right)(-1)^{(x-1)(n-1)/4}$.

Виконання описаних кроків неминуче зведе обчислення $\left(\frac{x}{n}\right)$ до обчислення символу Якобі вигляду $\left(\frac{1}{n'}\right)$, який за пунктом 5 твердження 1.7 дорівнює 1 для довільного непарного n'.

Приклад 1.8.

$$\left(\frac{12}{17}\right) = \left(\frac{2^2\cdot 3}{17}\right) = \left(\frac{2^2}{17}\right)\left(\frac{3}{17}\right) = \left(\frac{3}{17}\right) =$$
$$= \left(\frac{17 \bmod 3}{3}\right)(-1)^{(3-1)(17-1)/4} = \left(\frac{2}{3}\right) = (-1)^{(3^2-1)/8} = -1.$$

1.3. Розподіл простих чисел. Позначимо через $\pi(n)$ кількість простих чисел, що не перевищують n. Чебишов (1850) довів, що $0.92n/\ln n < \pi(n) < 1.1n/\ln n$, починаючи з деякого n. Адамар і Валле Пуссен (1896) довели, що $\frac{\pi(n)\ln n}{n}$ прямує до 1 при зростаючому n. В [75] наведено найліпші на цей час оцінки:

$$\pi(n) > \frac{n}{\ln n} \quad \text{для } x \geq 17,$$
$$\pi(n) < \frac{n}{\ln n - 4} \quad \text{для } x \geq 55.$$

Постулат Бертрана, доведений Чебишовим, говорить, що при $n > 2$ між n і $2n-2$ є хоча б одне просте число. Найкращий в цьому напрямку на сьогодні результат гарантує наявність простого числа між n і $n + n^{107/200}$ (див. монографію [75, стор. 225], а також огляд [42, стор. 62] і цитовану там літературу). Припущення, що для кожного $c > 2$ існує таке $d > 0$, що між n і $n + d(\ln n)^c$ є принаймні одне просте число, поки що не доведене і не спростоване.

ВПРАВИ

1.1. Знайти всі первісні корені за модулем $p = 3, 5, 7, 11$

a) безпосередніми обчисленнями згідно з означенням;

b) послуговуючись твердженням 1.2.

1.2. Скільки елементів якого порядку міститься в групі

a) $\mathbb{Z}_{23}^*$;

b) $\mathbb{Z}_p^*$ для такого простого p, що $p - 1 = 2q$ для деякого простого числа $q > 2$.

1.3. Довести, що для цілого числа g і простого p послідовність $g^i \bmod p$, де $i = 1, 2, 3, \ldots$, періодична. Якщо g не ділиться на p, то період цієї послідовності дорівнює порядку елемента $g \bmod p$ в групі $\mathbb{Z}_p^*$.

1.4. Нехай $g \in \mathbb{Z}_p^*$ — первісний корінь за простим модулем p. Довести, що відображення $f(x) = g^x$ є ізоморфізмом з адитивної групи $\mathbb{Z}_{p-1}$ на мультиплікативну групу $\mathbb{Z}_p^*$.

1.5. a) Припустимо, що g — первісний корінь за простим модулем p, і що числа k і $p-1$ взаємно прості. Довести, що степінь g^k також є первісним коренем за модулем p.

b) 2 є первісним коренем за модулем 13. Знайти всі інші первісні корені в $\mathbb{Z}_{13}^*$.

1.6. Виписати всі квадратичні лишки і нелишки за модулем $n = 3, 5, 7, 9, 11, 13, 15, 17$.

1.7. Обчислити символи Якобі a) $\left(\frac{79}{211}\right)$; b) $\left(\frac{113}{137}\right)$.

1.8. Довести, що описаний алгоритм обчислення символу Якобі $\left(\frac{x}{n}\right)$ затрачає не більше, ніж $2\log_2 n$ кроків.

1.9. a) Довести, що множина $\mathcal{Q}_n$ зведених квадратичних лишків за модулем n утворює підгрупу групи $\mathbb{Z}_n^*$.

b) Довести, що для простого p група $\mathcal{Q}_p$ є циклічною і має порядок $(p-1)/2$. (Таким чином, кількість квадратичних лишків і нелишків за простим модулем однакова.)

1.10. Нехай $n = pq$ — добуток двох різних непарних простих чисел.

a) Довести, що $\mathcal{Q}_n$ налічує $(p-1)(q-1)/4$ елементи.

b) Довести, що кожен елемент із $\mathcal{Q}_n$ має рівно 4 квадратні корені в $\mathbb{Z}_n^*$.

1.11. a) Задамо відображення $f : \mathbb{Z}_n^* \to \mathcal{Q}_n$ формулою $f(y) = y^2 \bmod n$. Довести, що f є гомоморфізмом груп.

b) Довести, що всі елементи із $\mathcal{Q}_n$ мають однакову кількість квадратних коренів в $\mathbb{Z}_n^*$, а саме $\|\mathbb{Z}_n^*\|/\|\mathcal{Q}_n\|$.

1.12. Довести, що для будь-якого модуля n

a) добуток квадратичних лишків є квадратичним лишком;

b) добуток квадратичного лишка і нелишка є квадратичним нелишком;

c) елемент, обернений до квадратичного лишка, є квадратичним лишком;

d) елемент, обернений до квадратичного нелишка, є квадратичним нелишком.

1.13. Нехай p — просте число. Довести, що

a) добуток двох квадратичних нелишків за модулем p є квадратичним лишком за цим модулем;

b) для будь-якого квадратичного нелишка z за модулем p, відображення $f_z(x) = (zx) \bmod p$ є бієкцією з множини $\mathcal{Q}_p$ квадратичних лишків на множину $\mathbb{Z}_p^* \setminus \mathcal{Q}_p$ квадратичних нелишків за цим модулем.

1.14. Нехай $n > 2$ непарне.

a) Довести, що відображення $f(x) = \left(\frac{x}{n}\right)$ є гомоморфізмом із групи $\mathbb{Z}_n^*$ в групу $\mathbb{Z}^* = \{\pm 1\}$.

b) Показати, що у випадку простого n ядром цього гомоморфізму є група $\mathcal{Q}_n$.

c) Довести, що $\mathbb{Z}_n^*$ містить однакову кількість елементів із символом Якобі 1 і -1.

d) Позначимо через $\mathbb{J}_n$ підмножину в $\mathbb{Z}_n^*$, яка складається з елементів із символом Якобі 1. Довести, що $\mathbb{J}_n$ є підгрупою в $\mathbb{Z}_n^*$.

1.15. Нехай

$$\widetilde{\mathcal{Q}}_n = \left\{ x \in \mathbb{Z}_n^* : \left(\frac{x}{n}\right) = 1, \text{ але } x \notin \mathcal{Q}_n \right\}.$$

Це так звана множина *псевдоквадратів* за модулем n. Припустимо, що $n = pq$ є добутком двох різних непарних простих. Довести, що

a) $\mathcal{Q}_n$ і $\widetilde{\mathcal{Q}}_n$ мають однакову кількість елементів;

b) для кожного $z \in \widetilde{\mathcal{Q}}_n$ відображення $f_z(x) = zx \bmod n$ є бієкцією із $\mathcal{Q}_n$ на $\widetilde{\mathcal{Q}}_n$.

1.16. Нехай p непарне просте. Довести, що

a) $\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}$;

b) конгруенція $x^2 \equiv -1 \pmod p$ має розв'язок для модулів вигляду $p = 4k+1$ і лише для них.

c) Довести рівність із пункту **a** для довільного непарного p.

1.17. Позначимо n-не просте число через $p(n)$: $p(1) = 2$, $p(2) = 3$, $p(3) = 5$ і т.д. З нерівностей пункту 1.3 вивести оцінки

$$n \ln n - O(1) < p(n) < n(\ln n + \ln \ln n + O(1)).$$

1.18. ([75]) Використовуючи оцінки попередньої задачі, довести, що задачі обчислення функцій $p(n)$ і $\pi(n)$ поліноміально еквівалентні.

ЛІТЕРАТУРА

Доведення квадратичного закону взаємності Гауса можна знайти в будь-якому із джерел [13, 2, 6]. Доведення оцінок Чебишова для функції $\pi(n)$ міститься у [15].

§ 2. Тестування простоти

Цей параграф присвячено задачі

ТЕСТУВАННЯ ПРОСТОТИ

Задано: $n \in \mathbb{N}$.

Розпізнати: чи n просте число?

Алгоритми розв'язування цієї задачі називаються *тестами простоти*. Якщо алгоритм приймає вхід n, то кажуть, що n *проходить* або *витримує* тест.

Перший тест простоти був винайдений понад два тисячоліття тому Ератосфеном з Александрії і називається *ситом Ератосфена*. Точніше, сито Ератосфена — це процедура, яка укладає список всіх простих чисел в межах від 1 до заданого n. Коли початковий відрізок простих

чисел вже знайдено, наступне просте визначається за таким критерієм: натуральне число є простим тоді і лише тоді, коли воно не ділиться на жодне із передуючих йому простих чисел. Наступний елементарний приклад тесту простоти спирається на ту ж ідею і вимагає часу не більше, ніж просіювання через сито Ератосфена.

Вхід: $n > 1$.

Присвоюємо змінній l значення 2.

- Якщо $l > \lfloor n/2 \rfloor$, то закінчуємо роботу і приймаємо вхід (тобто стверджуємо, що n просте).
- Якщо $l \le \lfloor n/2 \rfloor$ і $l \mid n$, то закінчуємо роботу і вхід відхиляємо (зауважимо, що при цьому знайдено перший простий дільник числа n).
- Якщо $l \le \lfloor n/2 \rfloor$ і $l \nmid n$, то збільшуємо l на 1 і вертаємось до виконання того ж умовного оператора.

Однак оцінювання часу роботи дає невтішний діагноз — описаний алгоритм експоненційний. Якщо вхід n є складеним числом, то перевірка цього може й не зайняти багато часу. Скажімо, якщо $n > 2$ парне, то це з'ясується вже на першому кроці. Але якщо вхід n є простим, то алгоритм змушений виконати $\lfloor n/2 \rfloor - 1$ крок для кожного l від 2 до $\lfloor n/2 \rfloor$, а це число є експонентою від довжини входу (див. обговорення на стор. 93).

Насправді можна обмежитись $\lfloor \sqrt{n} \rfloor$ кроками, позаяк якщо n має нетривіальний дільник l, то мусить мати і нетривіальний дільник, що не перевищує $\sqrt{n}$ (або сам l, або n/l). Це зауважив близько 1202 року Леонардо Пізанський, знаний як Фібоначчі. Втім, пришвидшений таким чином алгоритм залишається експоненційним. Наступне пониження оцінки часу, необхідного для тестування простоти, було зроблене лише у 1974 році в роботі [113], де запропоновано алгоритм із оцінкою часу $O(\sqrt[3]{n})$. Зазначимо, що цей алгоритм теж не тільки визначає, що число складене, а й знаходить його дільник.

Найкращий з існуючих детермінований алгоритм розроблено у [66]. Він розпізнає простоту за час $O((\log n)^{c \log\log\log n})$ для деякої константи c. Алгоритм з такою оцінкою часу роботи називають *квазіполіноміальним*. Що важливо, цей алгоритм є ефективним на практиці і реально розпізнає простоту чисел, які мають 100 цифр у десятковому записі.

Поліноміальних алгоритмів тестування простоти сьогодні невідомо.

2.1. Ймовірнісний тест Соловея-Штрассена. Ймовірнісний тест простоти був винайдений у [144]. Він є поліноміальним і має однобічну

помилку. Цей тест опирається на

Твердження 2.1. *Для непарного $n \geq 3$ покладемо*

$$S_n = \left\{ x \in \mathbb{Z}_n^* : \left(\frac{x}{n}\right) \equiv x^{(n-1)/2} \pmod n \right\}.$$

Якщо n складене, то $\|S_n\| \leq \phi(n)/2$.

Доведення. За мультиплікативністю символу Якобі (пункт 2 твердження 1.7) S_n є підгрупою мультиплікативної групи $\mathbb{Z}_n^*$. Щоб довести твердження, досить показати, що ця підгрупа є власною. Справді, за теоремою Лагранжа $\|S_n\|$ ділить $\|\mathbb{Z}_n^*\| = \phi(n)$, а тому, якщо ці числа не рівні між собою, то $\|S_n\|$ може бути щонайбільше $\phi(n)/2$.

Залишається довести існування елемента $y \in \mathbb{Z}_n^* \setminus S_n$. Розглянемо два випадки.

Випадок 1: n вільне від квадратів, тобто у розкладі $n = p_1 \dots p_k$ на прості множники кожне число зустрічається лише один раз.

Виберемо в $\mathbb{Z}_{p_1}^*$ квадратичний нелишок s за модулем p_1. За Китайською теоремою про остачі в $\mathbb{Z}_{p_1}^*$ існує y таке, що

$$y \equiv s \pmod{p_1}, \quad y \equiv 1 \pmod{p_2 \dots p_k}. \tag{1}$$

Для такого y за пунктами 4 і 1 твердження 1.7 маємо

$$\left(\frac{y}{n}\right) = \left(\frac{y}{p_1}\right)\left(\frac{y}{p_2 \dots p_k}\right) = \left(\frac{s}{p_1}\right)\left(\frac{1}{p_2 \dots p_k}\right) = (-1) \cdot 1 = -1.$$

Проте $y^{(n-1)/2} \not\equiv -1 \pmod n$, бо інакше було б $y^{(n-1)/2} \equiv -1 \pmod{p_2 \dots p_k}$, суперечність з (1).

Випадок 2: існує просте p таке, що $p^2 \mid n$.

Візьмемо $y = 1 + n/p$. Зрозуміло, що для кожного q, простого дільника числа n, виконується конгруенція $y \equiv 1 \pmod q$. Це означає, що $y \in \mathbb{Z}_n^*$ і $\left(\frac{y}{n}\right) = 1$, останнє за мультиплікативністю символу Якобі.

З іншого боку, $y^{(n-1)/2} \not\equiv 1 \pmod n$. Справді,

$$y^{(n-1)/2} = (1 + n/p)^{(n-1)/2} =$$
$$\sum_{i=0}^{(n-1)/2} \frac{\frac{n-1}{2}!}{i!(\frac{n-1}{2} - i)!}(n/p)^i \equiv 1 + \frac{n-1}{2} \cdot \frac{n}{p} \pmod n$$

(решта членів у біномі Ньютона кратні n). Отже, якби ліва частина дорівнювала 1 за модулем n, то ми мали б $\frac{n-1}{2} \cdot \frac{n}{p} \equiv 0 \pmod n$. Останнє можливе лише за умови, що $p \mid \frac{n-1}{2}$, що неможливо, бо $p \mid n$. ∎

Тест Соловея-Штрассена.

Вхід: n — непарне.

- Вибрати випадкове x таке, що $1 \leq x \leq n-1$.
- Якщо НСД $(x, n) \neq 1$, то припинити роботу з резолюцією "складене", інакше продовжувати.
- Якщо $\left(\frac{x}{n}\right) \not\equiv x^{(n-1)/2} \pmod n$, то зупинитись з резолюцією "складене", інакше зупинитись з резолюцією "просте".

Коректність. Тест стверджує, що вхід n є простим, для таких і лише таких x, що

$$\text{НСД}\,(x, n) = 1 \quad \text{і} \quad \left(\frac{x}{n}\right) \equiv x^{(n-1)/2} \pmod n.$$

Якщо n просте, то для всіх x від 1 до $n-1$ справедливі обидві рівності — перша за означенням простого числа, а друга за критерієм Ойлера. Отож просте n витримує тест з імовірністю 1.

Якщо n складене, то ці дві умови справедливі для всіх x з множини S_n, означеної у твердженні 2.1, і лише для них. Отже, складене n витримує тест з імовірністю $\frac{\|S_n\|}{n-1}$, що за твердженням 2.1 не перевищує $\frac{\phi(n)}{2(n-1)} < \frac{1}{2}$.

Таким чином, тест Соловея-Штрассена має однобічну помилку, меншу від $1/2$.

Ефективність. Перевірка всіх умов у тесті може бути здійснена ефективно. НСД (x, n) обчислюється за допомогою алгоритму Евкліда, $\left(\frac{x}{n}\right)$ — за допомогою алгоритму із пункту 1.2, а $x^{(n-1)/2} \bmod n$ — за допомогою бінарного методу з пункту III.2.2. Кожен з цих алгоритмів робить $O(\log n)$ операцій ділення з остачею чи множення в $\mathbb{Z}_n$. За умови, що ці операції виконуються за час $O(\log^2 n)$, час роботи тесту є $O(\log^3 n)$.

Ймовірність помилки можна знизити до 2^{-k}, повторюючи тест k разів згідно із загальною схемою, описаною в пункті III.3.1. Платою за це буде збільшення часу роботи у k разів.

2.2. Лас Вегас алгоритм. Тест Соловея-Штрассена розпізнає множину простих чисел за поліноміальний час з однобічною помилкою. В [65] запропоновано ймовірнісний поліноміальний алгоритм, що розпізнає з однобічною помилкою множину складених чисел. Як наслідок, для тестування простоти існує Лас Вегас алгоритм (див. вправу III.3.3).

2.3. Псевдопрості числа. Мала теорема Ферма стверджує, що якщо n — просте число, то для всіх $x \in \mathbb{Z}_n^*$

$$x^{n-1} \equiv 1 \pmod{n}. \tag{1}$$

Непарне n, яке задовольняє умову (1), але не є простим, називається *псевдопростим числом Ферма* або просто *псевдопростим за основою* x. Деякі складені числа є псевдопростими за будь-якою основою. Це так звані *числа Кармайкла*, найменшим з яких є $561 = 3 \cdot 11 \cdot 17$.

Критерій Ойлера стверджує, що якщо n — непарне просте число, то для всіх $x \in \mathbb{Z}_n^*$ виконується конгруенція

$$\left(\frac{x}{n}\right) \equiv x^{(n-1)/2} \pmod{n}. \tag{2}$$

Непарне n, яке задовольняє умову (2), але не є простим, називається *псевдопростим числом Ойлера за основою* x.

Оскільки із (2) випливає (1), то кожне псевдопросте число Ойлера за основою x є також псевдопростим числом Ферма за цією ж основою. Зворотне твердження неправильне. З нього випливало б, що кожне число Кармайкла є псевдопростим числом Ойлера за будь-якою основою. Однак чисел з такою властивістю не існує за твердженням 2.1.

Знову припустимо, що n непарне просте. Нехай $n - 1 = 2^s t$, де t непарне. Для елемента x групи $\mathbb{Z}_n^*$ розглянемо в цій групі послідовність

$$x^t, x^{2t}, x^{4t}, \ldots, x^{2^s t}.$$

Останній член цієї послідовності дорівнює 1 за малою теоремою Ферма, а кожен попередній є квадратним коренем наступного. Оскільки $\mathbb{Z}_n^*$ для простого n є полем, то коренем з 1 може бути лише 1 або -1. Отже, або вся послідовність складається з одиниць, або має хвіст із деякого числа одиниць, перед якими знаходиться -1. Остання умова рівносильна такій:

$$\begin{gathered} \text{або } x^t \equiv 1 \pmod{n}, \\ \text{або існує } j,\ 0 \leq j \lneq s, \text{ таке, що } x^{2^j t} \equiv -1 \pmod{n}. \end{gathered} \tag{3}$$

Непарне n, де $n - 1 = 2^s t$ з непарним t, яке задовольняє умові (3) для $x \in \mathbb{Z}_n^*$ називається *сильно псевдопростим числом за основою* x.

Доведення наступних двох тверджень можна знайти в [111, твердження V.1.6] або [55, теорема 17.2], і в [111, твердження V.1.7], відповідно.

Твердження 2.2. *З умови (3) випливає умова (2). Отже, сильно псевдопрості числа за основою x є псевдопростими числами Ойлера за цією ж основою.* ∎

Твердження 2.3. *Непарне складене число n є сильно псевдопростим за основою x щонайбільше для четвертої частини всіх x таких, що $1 \leq x < n$.* ∎

2.4. Ймовірнісний тест Міллера-Рабіна. На твердженні 2.3 грунтується

Тест Міллера-Рабіна.

Вхід: n — непарне натуральне число, $n - 1 = 2^s t$, t непарне.

- Вибрати випадкове x таке, що $1 \leq x < n$.
- Якщо НСД$(x, n) \neq 1$, то зупинитись з резолюцією "n складене", інакше продовжувати.
- Обчислити $y_0 = x^t \pmod{n}$.
- Якщо $y_0 \equiv \pm 1 \pmod{n}$, то зупинитись з резолюцією "просте", інакше продовжувати.
- Обчислювати $y_j = y_{j-1}^2 \pmod{n}$, доки не виявиться, що $y_j \equiv \pm 1 \pmod{n}$.
- Якщо $y_j \equiv 1 \pmod{n}$, то зупинитись з резолюцією "складене", а якщо $y_j \equiv -1 \pmod{n}$, то зупинитись з резолюцією "просте".

Час роботи цього тесту $O(\log^3 n)$. З твердження 2.3 і обговорення перед ним безпосередньо випливає, що помилка тесту однобічна, з імовірністю щонайбільше $1/4$. Повторенням тесту k разів помилка зменшується до $(1/4)^k$. Твердження 2.2 показує, що тест Міллера-Рабіна є формальним підсиленням тесту Соловея-Штрассена із вдвічі кращою оцінкою ймовірності помилки.

2.5. Прості числа до 25 000 000 000. В [128] доведено, що серед чисел, менших від $25 \cdot 10^9$, лише одне є сильно псевдопростим за кожною із чотирьох основ 2, 3, 5, 7. Цим числом є 3 215 031 751. Отже, дуже легко перевірити, чи задане число $n < 25 \cdot 10^9$ є простим. Для цього необхідно і досить, щоб n було відмінним від вказаного вище складеного числа, і щоб умова (3) виконувалась для $x = 2, 3, 5, 7$.

2.6. Розширена гіпотеза Рімана. *Дзета функція Рімана* є функцією від комплекснозначної змінної $s = \sigma + i\tau$, яка означується за допо-

могою ряду

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}.$$

Цей ряд абсолютно збігається у будь-якій області з $\sigma \geq 1$, і його сума є аналітичною функцією, що аналітично продовжується на всю комплексну площину з єдиною особливістю в точці $s = 1$. Недоведена досі *гіпотеза Рімана* стверджує, що всі нулі дзета-функції в смузі $0 \leq \sigma \leq 1$ лежать на прямій $\sigma = 1/2$.

Характером групи $\mathbb{Z}_m^*$ називається довільний її гомоморфізм у мультиплікативну групу комлексних чисел $\mathbb{C}^*$. Кожен характер $g : \mathbb{Z}_m^* \to \mathbb{C}^*$ визначає функцію $\chi : \mathbb{N} \to \mathbb{C}^*$,

$$\chi(n) = \begin{cases} g(n \bmod m), & \text{якщо НСД}\,(n, m) = 1, \\ 0, & \text{якщо НСД}\,(n, m) \neq 1, \end{cases}$$

яка називається *характером Діріхле за модулем* m. Характер називається головним, якщо він приймає значення лише 0 та 1. Для характера Діріхле χ, *L-функція Діріхле* від комплексної змінної s означується як сума ряду

$$L(s, \chi) = \sum_{n=1}^{\infty} \chi(n) n^{-s},$$

який збігається в області $\sigma > 0$ (див. [15]).

Розширена гіпотеза Рімана, для якої далі буде вживатися абревіатура РГР, стверджує, що для неголовного характеру χ всі нулі функції $L(s, \chi)$ у півплощині $\sigma > 0$ знаходяться на прямій $\sigma = 1/2$.

За припущення справедливості РГР Міллер довів, що кожне непарне складене n принаймні для однієї основи $x < 2(\ln n)^2$ не є сильно псевдопростим числом за цією основою. Звідси негайно випливає, що за умови справедливості РГР для тестування простоти існує детермінований поліноміальний алгоритм. Цей алгоритм констатує простоту заданого непарного n в разі виконання умови (3) для всіх $x < 2(\ln n)^2$. Зауважимо, що від справедливості РГР залежить не поліноміальність цього алгоритму, яка є безумовним фактом, а його коректність. Якщо РГР хибна, то не виключено, що описаний тест витримують деякі складені числа.

2.7. Виробництво простих чисел. Багато криптосистем використовують параметр, який є простим числом, причому надійність криптосистеми грунтується на тому, що цей параметр тримається від

суперника в таємниці. Щоб позбавити суперника найменших шансів відгадати це просте число, останнє слід вибирати великим і випадковим. Зокрема, не слід користуватись простими числами спеціального виду на зразок простих чисел Мерсенна (див. вправу 2.7), хоча вони і приваблюють добре вивченими критеріями простоти. Вибір простих чисел гарантується в широкому асортименті оцінкою Чебишова для функції $\pi(n)$ та пізнішими результатами про розподіл простих чисел, цитованими у розділі 1.3.

Сучасні криптографічні стандарти вимагають простих чисел, що мають близько сотні десяткових цифр. Цей розмір диктується тим, що добуток таких чисел сьогодні важко розкласти на множники (див. наступний параграф).

Генерування випадкового простого числа заданого розміру l відбувається наступним чином. Вибирається випадкове натуральне число n між 10^{l-1} і 10^l. Далі почергово перебираються числа $n+1$, $n+2$, $n+3$ і т.д., кожне з яких випробовується тестом простоти. Зауважимо, що числа, кратні кільком першим простим, вигідніше відкидати негайно, а не в результаті тестування (за ознаками подільності на 2, 3, 5, 7, 11, 13, 17 і т.д. звертатись до [13]). Цей процес триває до того моменту, поки чергове число $n + m$ не витримає тест. Воно й береться в якості шуканого простого.

Слід звернути увагу на такі нюанси. Якщо застосовується ймовірнісний тест простоти з однобічною помилкою, то вибране число $n + m$ є простим не напевне, а з великою достовірністю. А саме, якщо тест k-кратний, то ймовірність того, що число насправді складене, є експоненційно низькою, наприклад 2^{-k} у випадку тесту Соловея-Штрассена. Якщо наявні обчислювальні потужності дозволяють взяти $k = 100$, то ймовірність помилитися стає цілком прийнятною, адже вона не набагато більша за ймовірність того, що суперник зламає криптосистему, банальним чином відгадавши таємне просте число. Числа, які вірогідно, але не напевне прості, часом називають *комерційними*.

Зупинимось на питанні, якого часу вимагає описана процедура. Зрозуміло, що він обмежується величиною $O(m \cdot t(n+m))$, де $n+m$ — перше просте число після n, а $t(z)$ — час роботи на вході z тесту простоти, що використовується. Відомі верхні оцінки на $m = m(n)$ наведено у пункті 1.3. Хоча справедливість гіпотези $m(n) = (\log n)^{O(1)}$ невідома, наступні евристичні міркування не розходяться з емпіричним досвідом. Асимптотичний закон розподілу простих чисел (див.

пункт 1.3) дає підстави сподіватися, що від $n \approx 10^{100}$ найближче просте знаходиться на відстані $m \approx \ln(10^{100}) < 230$. Якщо ми заощаджуємо тести простоти на числах кратних 2, 3 і 5, то належить протестувати не більше, ніж $(\frac{1}{2}+\frac{1}{3}+\frac{1}{5}-\frac{1}{2\cdot3}-\frac{1}{2\cdot5}-\frac{1}{3\cdot5}+\frac{1}{2\cdot3\cdot5})\cdot 230 < 170$ чисел. Якщо виключити й числа кратні 7 і 11, то ця кількість зменшиться до 150.

Часто в застосуваннях з тою чи іншою метою виникає потреба в простому числі p, яке володіє додатковими властивостями. Наприклад, часом бажано, щоб $p-1$ мало великий простий дільник, або щоб був відомий розклад числа $p-1$ на прості співмножники. Параграфи, присвячені подібним застосуванням, будуть містити посилання на літературу з відповідними алгоритмами.

ВПРАВИ

2.1. Перевірити, чи є 91 за основою 3 псевдопростим числом **a)** Ферма, **b)** Ойлера.

Наступні три вправи запозичені з [111].

2.2. Знайти всі основи x, за якими числа **a)** 15, **b)** 21 є псевдопростими.

2.3. Знайти найменше псевдопросте за основою **a)** 5, **b)** 2.

2.4. Нехай $n = pq$ — добуток двох різних простих чисел, і $d = \text{НСД}(p-1, q-1)$. Довести, що n псевдопросте за основою x тоді і лише тоді, коли $x^d \equiv 1 \pmod n$. Виразити через d кількість основ, за якими n псевдопросте.

2.5. Довести, що якщо n є псевдопростим числом Ойлера і $n \equiv 3 \pmod 4$, то n є сильно псевдопростим.

2.6. a) Припустимо, що число n псевдопросте за основами x і y. Довести, що воно псевдопросте також за основою $(xy) \bmod n$.

b) Припустимо, що n є псевдопростим числом Ойлера за основами x і y. Довести це також для основи $(xy) \bmod n$.

c) Перевірити, що 65 є сильно псевдопростим за основами 8 і 18, але не за основою $14 = (8 \cdot 18) \bmod 65$.

2.7. a) Довести, що якщо число $2^m - 1$ просте, то m також мусить бути простим. Прості числа такого виду називаються простими Мерсенна.

b) Довести, що якщо число $2^m + 1$ просте, то m мусить бути степенем числа 2. Прості числа такого виду називаються простими Ферма.

Зауваження. Досьогодні невідомо, чи простих Мерсенна є безліч. На 1995 рік було знайдено 33 значення m, для яких $2^m - 1$ є простим, найбільше з яких є $m = 859\,433$ ([127]). Першим складеним числом вигляду $2^m - 1$ для простого m є $2^{11} - 1 = 23 \cdot 89$. Евклід зауважив, що якщо $2^m - 1$ є (в сучасній термінології) простим Мерсенна, то число $2^m(2^m - 1)$ є *досконалим*, тобто воно дорівнює сумі всіх своїх дільників (за винятком себе самого). Ойлер довів, що кожне парне досконале число має саме такий вигляд. Питання про існування непарних досконалих чисел відкрите.

Першим складеним числом вигляду $2^{2^l}+1$ є $2^{2^5}+1$. Розклад цього числа на множники, $4294967297 = 641 \cdot 6700417$, отримав Ойлер, спростувавши гіпотезу Ферма, що всі подібні числа є простими. В той час як для багатьох чисел вигляду $2^{2^l}+1$ з $l > 5$ показано, що вони складені, простоту жодного такого числа досі не встановлено [55].

Критерії простоти для чисел Мерсенна і Ферма описані в [55, 12].

2.8. Розглянемо задачу

ПОШУК ПРОСТОГО ПОЗА ЗАДАНОЮ МЕЖЕЮ

Задано: $n \in \mathbb{N}$.

Знайти: просте $p \geq n$.

a) Придумати зведення цієї задачі до тестування простоти, яке було б поліноміальним за умови справедливості згаданої в пункті 1.3 гіпотези, що $p - n = (\log n)^{O(1)}$ для простого числа p, найближчого до n зверху. Використовуючи це зведення, описати детермінований алгоритм для пошуку $p \geq n$, який би був поліноміальним за умови, що справедлива РГР.

b) Придумати поліноміальне ймовірнісне зведення цієї задачі до тестування простоти, і на його основі розробити поліноміальний ймовірнісний алгоритм.

ЛІТЕРАТУРА

В [68] встановлено, що Кармайклових чисел є безліч. Про детермінований тест простоти з [66] йдеться також в [38, 33, 12, 42]. Огляди інших тестів зроблено в [12, 55, 114, 32].

§ 3. Факторизація

Сформулюємо задачу *факторизації* або *розкладу на прості множники* натурального числа.

ФАКТОРИЗАЦІЯ

Задано: $n \in \mathbb{N}$ $(n > 1)$.

Знайти: прості $p_1, \ldots, p_s$ і натуральні $\alpha_1, \ldots, \alpha_s$ такі, що $n = \prod_{i=1}^{s} p_i^{\alpha_i}$.

Зазначимо, що факторизація натурального числа, тобто задача пошуку *всіх* його простих дільників, еквівалентна задачі пошуку *хоча б одного* простого дільника (див. приклад III.4.3). Слід також відзначити, що ефективні тести простоти із попереднього параграфу хоча й дозволяють швидко розпізнати, складеним чи простим є задане число, але не дають жодної інформації про дільники складеного числа.

На сьогоднішній день задача факторизації є важкою. Найефективніші з відомих алгоритмів факторизації, включно з ймовірнісними, потребують часу

$$\exp\left\{c\sqrt{\ln n \ln\ln n}\right\}, \tag{1}$$

причому у найкращому випадку $c = 1$. Є також алгоритми з оцінкою часу

$$\exp\left\{c(\ln n)^{1/3}(\ln\ln n)^{2/3}\right\}, \tag{2}$$

яка не доведена, а припускається на основі деяких евристичних міркувань.

На межі сучасних можливостей є факторизація чисел із 150 десятковими цифрами. Розклад на множники чисел, які мають 200 десяткових цифр, на думку експертів, залишається справою майбутнього.

Для зламу деяких криптосистем суперникові досить розкласти на множники число n, про яке відомо, що воно є добутком двох простих. Тобто виникає звужена задача факторизації.

Задано: $n = pq$, де p і q прості.
Обчислити: p і q.

Для звуженої задачі невідомо нічого кращого, ніж у загальному випадку. Однак деякі алгоритми факторизації досягають успіху, якщо p і q задовольняють певні умови. Сформульовано властивості, які виключають виконання цих небажаних умов. Прості числа, що володіють цими властивостями, називаються *сильно простими*. Спосіб їх породження запропоновано в [102].

Цікаво також зауважити, що для розпізнавання, чи натуральне число є добутком рівно двох простих, невідомо кращого способу, ніж безпосередня факторизація.

ВПРАВИ

3.1. a) Розробити метод факторизації чисел вигляду $n = pq$, де p і q — прості числа-близнята, тобто $q = p + 2$.

b) Розробити метод факторизації чисел вигляду $n = pq$ для простих p і q, ефективний у випадку, коли різниця $q - p$ невелика. (Спосіб, викладений у розділі розв'язків, називається *факторизацією Ферма*.)

3.2. ([111]) a) Нехай для n відомо таке $x \in \mathbb{Z}_n^*$, що n є псевдопростим, але не сильно псевдопростим за основою x. Вказати спосіб швидкого знаходження нетривіального дільника числа n.

b) Пояснити, як слід вибирати прості p і q, щоб число $n = pq$ було важко факторизувати шляхом знаходження x як у пункті b.

ЛІТЕРАТУРА

За детальнішою інформацією про алгоритми факторизації слід звертатися до оглядів [42, 114, 64], підручника [111] та монографії [32, розділ 4.5.4].

§ 4. Розпізнавання квадратичності і добування квадратних коренів

4.1. Квадратичність у випадку простого модуля. Насамперед ми розглянемо задачу

РОЗПІЗНАВАННЯ КВАДРАТИЧНИХ ЛИШКІВ ЗА ПРОСТИМ МОДУЛЕМ
Задано: x, $p \in \mathbb{N}$, де $p > 2$ просте і $p \nmid x$.
Розпізнати: чи існує y таке, що $x \equiv y^2 \pmod p$?

Очевидно, що задача еквівалентна обчисленню символа Лежандра $\left(\frac{x}{p}\right)$, яке можна виконати двома способами. Перший полягає у використанні критерію Ойлера, за яким

$$\left(\frac{x}{p}\right) \equiv x^{(p-1)/2} \pmod p.$$

При цьому права частина ефективно обчислюється за допомогою бінарного методу піднесення у степінь за модулем (пункт III.2.2). Другий спосіб полягає у трактуванні $\left(\frac{x}{p}\right)$ як символу Якобі і використанні алгоритму із пункту 1.2 для обчислення останнього.

Нагадаємо, що через $\mathcal{Q}_p$ позначається множина зведених квадратичних лишків, тобто множина квадратичних лишків в $\mathbb{Z}_p^*$. Випадковий елемент множини $\mathcal{Q}_p$ породжується легко — досить взяти випадковий елемент в $\mathbb{Z}_p^*$ і піднести його до квадрату за модулем p. Дещо складніше вибрати в $\mathbb{Z}_p^*$ випадковий квадратичний нелишок. Для цього вибираємо випадковий елемент в $\mathbb{Z}_p^*$ і перевіряємо одним із двох згаданих вище способів, є він нелишком, чи ні. Якщо нам не пощастило, то ще раз вибираємо елемент в $\mathbb{Z}_p^*$ випадково і незалежно від попереднього вибору, і перевіряємо, чи не виявиться нелишком він. Так продовжуємо, поки не натрапимо на нелишок. Оскільки квадратичних лишків та нелишків в $\mathbb{Z}_p^*$ однакова кількість (див. вправу 1.9), то ймовірність вибору нелишка дорівнює $1/2$. Отже, ймовірність того, що ми не знайдемо нелишок упродовж k ітерацій, дорівнює 2^{-k}. Обчислення, подібні

до проведених у пункті III.3.2, показують, що математичне сподівання кількості потрібних ітерацій дорівнює 2.

Тепер сформулюємо задачу знаходження квадратичного нелишка (не обов'язково випадкового) в явному вигляді.

Знаходження квадратичного нелишка за простим модулем

Задано: просте p.

Знайти: $x \in \mathbb{Z}_p^* \setminus Q_p$.

Ми вже описали для цієї задачі поліноміальний Лас Вегас алгоритм із ймовірністю невдачі $1/2$, яку можна зменшити до 2^{-k} повторенням алгоритму k разів. Детермінованого поліноміального алгоритму для цієї задачі невідомо.

4.2. Добування квадратного кореня за простим модулем. Наступною ми розглянемо задачу

Добування квадратного кореня за простим модулем

Задано: $x, p \in \mathbb{N}$, де $p > 2$ просте і $p \nmid x$.

Знайти: y таке, що $x \equiv y^2 \pmod p$, якщо існує.

Для цієї задачі поліноміальний детермінований алгоритм відомий лише за справедливості РГР. Нижче подається поліноміальний ймовірнісний алгоритм.

Оскільки квадратичність за простим модулем розпізнається ефективно, ми можемо обмежитись випадком, коли x є квадратичним лишком за модулем p. У наступних двох часткових випадках задача легко розв'язується детермінованим чином.

Випадок 1: $p = 4m + 3$.

За критерієм Ойлера маємо $x^{2m+1} \equiv 1 \pmod p$, звідки $(x^{m+1})^2 \equiv x \pmod p$. Таким чином, можна взяти $y = \pm x^{m+1} \pmod p$ (це обидва корені з x в $\mathbb{Z}_p^*$).

Випадок 2: $p = 8m + 5$.

За критерієм Ойлера маємо $x^{4m+2} \equiv 1 \pmod p$, звідки $x^{2m+1} \equiv \pm 1 \pmod p$ і $x^{2m+2} \equiv \pm x \pmod p$. Який саме знак має місце у правій частині останнього порівняння, можна визначити безпосереднім обчисленням $x^{2m+1} \bmod p$ за допомогою бінарного методу. Якщо права частина виявиться рівною $+1$, то діємо як у випадку 1. Якщо ж у правій частині маємо -1, то використовуємо пункт 2 квадратичного закону взаємності Гауса, за яким $\left(\frac{2}{p}\right) = -1$ і, знову за критерієм Ойлера, $2^{4m+2} \equiv -1 \pmod p$. Тому маємо $x^{2m+2}2^{4m+2} \equiv x \pmod p$, і $y = \pm x^{m+1}2^{2m+1} \bmod p$.

Залишився

Випадок 3: $p = 8m + 1$.

Опишемо, як у цьому випадку добувати квадратний корінь, якщо додатково відомо квадратичний нелишок a за модулем p. Цим ми зведемо нашу задачу до знаходження квадратичного нелишка за простим модулем, для чого у попередньому пункті була вказана ймовірнісна процедура. Це єдине місце в алгоритмі, де використовується випадковість.

Нехай $p = 2^l h + 1$, де $l \geq 3$ і h непарне. За критерієм Ойлера маємо $x^{2^{l-1}h} \equiv 1 \pmod p$, звідки $x^{2^{l-2}h} \equiv \pm 1 \pmod p$. Застосувавши критерій Ойлера до квадратичного нелишка a, отримаємо $a^{2^{l-1}h} \equiv -1 \pmod p$. З останніх двох порівнянь для деякого цілого s_1, рівного 0 або h, випливає

$$x^{2^{l-2}h}a^{s_1 2^{l-1}} \equiv 1 \pmod p, \quad x^{2^{l-3}h}a^{s_1 2^{l-2}} \equiv \pm 1 \pmod p.$$

З останнього порівняння таким же чином для деякого цілого невід'ємного s_2 отримуємо

$$x^{2^{l-3}h}a^{s_2 2^{l-2}} \equiv 1 \pmod p, \quad x^{2^{l-4}h}a^{s_2 2^{l-3}} \equiv \pm 1 \pmod p.$$

Повторивши подібну процедуру ще $l-3$ рази, приходимо до порівняння $x^h a^{2s_{l-1}} \equiv 1 \pmod p$ для деякого цілого невід'ємного s_{l-1}, звідки остаточно отримуємо

$$y = \pm x^{(h+1)/2}a^{s_{l-1}} \bmod p.$$

4.3. Випадок модуля $n = pq$. Цей пункт присвячено двом задачам.

Розпізнавання квадратичних лишків за модулем $n = pq$

Задано: $x, n \in \mathbb{N}$, $n = pq$ для різних непарних простих p і q, НСД $(x, n) = 1$.

Розпізнати: чи існує y таке, що $x \equiv y^2 \pmod n$?

Добування квадратного кореня за модулем $n = pq$

Задано: $x, n \in \mathbb{N}$, $n = pq$ для різних непарних простих p і q, НСД $(x, n) = 1$.

Знайти: y таке, що $x \equiv y^2 \pmod n$, якщо існує.

Слід підкреслити, що співмножники p і q не задаються в умові задачі. За умовою відомо лише, що n є добутком двох простих чисел, але самі ці числа невідомі, а їх визначення за n є важкою задачею (див. § 3).

Твердження 4.1. *Нехай $n = pq$ є добутком двох різних непарних простих чисел. Припустимо, що число x взаємно просте з n, і позначимо $x_1 = x \bmod p$, $x_2 = x \bmod q$. Тоді мають місце такі факти.*

1) *Якщо y задовольняє конгруенцію*

$$y^2 \equiv x \,(\mathrm{mod}\, n), \tag{1}$$

то лишки

$$y_1 = y \bmod p, \quad y_2 = y \bmod q \tag{2}$$

задовольняють конгруенції

$$y_1^2 \equiv x_1 \,(\mathrm{mod}\, p), \quad y_2^2 \equiv x_2 \,(\mathrm{mod}\, q). \tag{3}$$

Навпаки, якщо y_1 та y_2 задовольняють конгруенції (3), то кожне y, що задовольняє (2), задовольняє також (1).

2) *x є квадратичним лишком за модулем n тоді і лише тоді, коли x є квадратичним лишком за кожним із модулів p і q.*

3) *Якщо x є квадратичним лишком за модулем n, то конгруенція (1) має в $\mathbb{Z}_n^*$ рівно 4 різних розв'язки: y з лишками (y_1, y_2), $-y$ з лишками $(p-y_1, q-y_2)$, y' з лишками $(y_1, q-y_2)$, та $-y'$ з лишками $(p-y_1, y_2)$.*

Доведення. Пункт 1 випливає безпосередньо із наслідку II.2.13. Пункт 2 є наслідком пункту 1.

Залишилось обгрунтувати пункт 3. За пунктом 1 конгруенція (1) має стільки ж розв'язків, скільки їх має система (3). Кожна із конгруенцій (3) має рівно 2 взаємно протилежні розв'язки, а відтак саму систему (3) задовольняють рівно 4 різні пари лишків: (y_1, y_2), $(p-y_1, q-y_2)$, $(y_1, q-y_2)$ та $(p-y_1, y_2)$. ∎

Як показує твердження 4.1, щоб розв'язати конгруенцію (1) (або розпізнати її розв'язність), досить зробити це для кожної із конгруенцій (3). Тобто, від модуля $n = pq$ ми переходимо до простих модулів p і q. А для простих модулів задачі, що розглядаються, мають ефективні алгоритми, описані у попередніх пунктах. Слід підкреслити, що перехід від модуля n до модулів p і q можливий лише за умови, що ми здатні розкласти число $n = pq$ на співмножники. Таким чином, задачі розпізнавання квадратичності та добування квадратного кореня за модулем $n = pq$ зводяться до факторизації чисел вигляду $n = pq$.

Зупинимось на цих зведеннях дещо детальніше. Припустимо, що нам відомі обидва дільники p і q модуля n. За пунктом 2 твердження 4.1 число x є квадратичним лишком за модулем n тоді і лише тоді, коли x є квадратичним лишком за кожним із модулів p і q, що перевіряється якимось із двох способів, згаданих у пункті 4.1.

Для того ж, щоб добути з x квадратний корінь за модулем n, тобто розв'язати конгруенцію (1), можна діяти наступним чином. Слід розв'язати обидві конгруенції (3) за допомогою алгоритму із пункту 4.2, отримавши в результаті y_1 і y_2. Після цього квадратний корінь y визначається з умов (2), як описано у пункті II.2.4 (приклад II.2.11).

Зауваження 4.2. Описане зведення добування кореня за модулем $n = pq$ до факторизації числа n є ймовірнісним, оскільки таким є алгоритм добування коренів y_1 і y_2 за простими модулями p і q. Однак у частковому випадку, коли $p \equiv q \equiv 3 \,(\mathrm{mod}\, 4)$ для добування квадратного кореня за модулями p і q є ефективна детермінована процедура (див. випадок 1 алгоритму із пункту 4.2), яка в цьому випадку дає ефективну детерміновану процедуру добування кореня і за модулем n.

Приклад 4.3. Обчислимо квадратні корені з $x = 58$ за модулем $n = 77$ ($p = 7$ і $q = 11$). Спершу обчислюємо $x_1 = 58 \bmod 7 = 2$ і $x_2 = 58 \bmod 11 = 3$. Використовуємо алгоритм добування коренів за простим модулем з пункту 4.2, причому маємо справу з випадком 1. Знаходимо $y_1 = 2^{(7-3)/4+1} \bmod 7 = 4$ і $y_2 = 3^{(11-3)/4+1} \bmod 11 = 5$. За алгоритмом, що міститься в доведенні Китайської теореми про остачі (див. приклад II.2.11), отримуємо $y = 60$. Для пари $(-y_1, y_2) = (3, 5)$ таким же чином знаходимо ще один корінь $y' = 38$. Ще два корені 17 і 39 протилежні до вже знайдених.

Виявляється, що й, навпаки, факторизацію чисел вигляду $n = pq$ можна звести до задачі знаходження (хоча б одного) квадратного кореня за модулем $n = pq$ за допомогою ефективної ймовірнісної процедури. Таким чином, задачі добування хоча б одного кореня, знаходження всіх коренів, і факторизації є попарно еквівалентними відносно поліноміальної ймовірнісної звідності. Зведення, яке ми опишемо, грунтується на такому спостереженні.

Твердження 4.4. *Нехай y та y' — два квадратні корені із числа x за модулем $n = pq$, де p і q різні непарні прості, жодне з яких не ділить x. Припустимо також, що*

$$y \not\equiv \pm y' \,(\mathrm{mod}\, n). \tag{4}$$

Тоді НСД $(y + y', n)$ *є одним із дільників p або q числа n.*

Доведення. За умовою маємо $y^2 \equiv (y')^2 \equiv x \pmod n$, звідки $y^2 - (y')^2 \equiv 0 \pmod n$ і

$$(y + y')(y - y') \equiv 0 \pmod n. \tag{5}$$

З умови (4) випливає, що ні $y + y'$, ні $y - y'$ не діляться націло на n. З врахуванням цього, співвідношення (5) дає, що НСД$(y + y', n)$ не дорівнює ні 1, ні n. Відтак НСД$(y + y', n)$ дорівнює або p, або q. ▪

Припустимо тепер, що оракул $\mathcal{O}$, отримавши квадратичний лишок x за модулем $n = pq$, видає якийсь квадратний корінь з x за модулем n. Наступний Лас Вегас алгоритм з доступом до оракула $\mathcal{O}$ знаходить нетривіальний дільник числа n.

Вхід: $n = pq$, де $p \neq q$ непарні прості.

- Вибрати випадковий елемент y в $\mathbb{Z}_n^*$. Обчислити $x = y^2 \bmod n$.
- Зробити оракулу $\mathcal{O}$ запит x. Нехай $y' = \mathcal{O}(x)$ — відповідь оракула, $(y')^2 \equiv x \pmod n$.
- Якщо виконується умова (4), то обчислити за допомогою алгоритму Евкліда НСД$(y + y', n)$ і результат подати на вихід.

Якщо справджується умова (4), то описаний алгоритм згідно із твердженням 4.4 справді дає нетривіальний дільник числа n. Оскільки y вибирається випадково, то за пунктом 3 твердження 4.1 умова (4) виконується з імовірністю 1/2. Якщо виконувати алгоритм k разів, то ймовірність того, що подія (4) не відбудеться жодного разу, стане 2^{-k}.

Зауваження 4.5. Як зазначалось у пункті 1.2, якщо x — квадратичний лишок за модулем n, то $\left(\frac{x}{n}\right) = 1$, хоча обернене твердження справедливе не завжди. Оскільки символ Якобі легко обчислюється, то задача розпізнавання квадратичних лишків за модулем $n = pq$ еквівалентна такому своєму звуженню.

Задано: $x, n \in \mathbb{N}$, $n = pq$ — добуток непарних простих $p \neq q$, НСД$(x, n) = 1$, $\left(\frac{x}{n}\right) = 1$.
Розпізнати: чи існує y таке, що $x \equiv y^2 \pmod n$?

Як було показано, ця задача не важча, ніж факторизація модуля $n = pq$. Однак ніякого ефективного алгоритму на сьогоднішній день для цієї задачі невідомо.

ВПРАВИ

4.1. a) Довести, що якщо y є випадковим елементом в $\mathbb{Z}_n^*$, де n просте, то $x = y^2 \bmod n$ є випадковим елементом із $\mathcal{Q}_n$.

b) Те ж саме для $n = pq$, де p і q різні прості.

c) Те ж саме для довільного n.

4.2. Чи є квадратичним лишком за модулем 37 число a) 15, b) 30?

4.3. Знайти якийсь квадратичний нелишок за модулем 83.

4.4. Застосовуючи алгоритм добування квадратного кореня за простим модулем, знайти a) $\sqrt{5} \bmod 19$; b) $\sqrt{5} \bmod 29$; c) $\sqrt{2} \bmod 41$. В останньому пункті використати факт, що 6 є квадратичним нелишком за модулем 29.

4.5. Знайти всі квадратні корені з 60 за модулем 77.

4.6. Нехай $n = pq$, де $p \neq q$ прості. Згідно з пунктом 3 твердження 4.1 кожен квадратичний лишок за модулем n має рівно 4 квадратні корені за цим модулем. Довести, що задача знаходження всіх коренів з числа за заданим одним із них еквівалентна факторизації модуля n відносно ймовірнісної поліноміальної звідності.

4.7. Число $n = pq$ називається *цілим Блюма*, якщо p і q — різні прості з властивістю $p \equiv q \equiv 3 \pmod 4$. Розглядаємо звуження задачі розпізнавання квадратичних лишків на множину модулів n, які є цілими Блюма. Припустимо, що існує поліноміальний ймовірнісний алгоритм A, який, отримавши на вхід випадковий елемент x множини $\mathbb{J}_n = \mathcal{Q}_n \cup \widetilde{\mathcal{Q}}_n$, розпізнає, чи $x \in \mathcal{Q}_n$, з ймовірністю щонайменше $1/2 + 1/(\log n)^c$ для деякої константи c, де ймовірність визначається як випадковим вибором x, так і випадковою послідовністю алгоритму. Довести, що тоді квадратичні лишки (за модулями, які є цілими Блюма) розпізнаються за поліноміальний час деяким ймовірнісним алгоритмом A'.

ЛІТЕРАТУРА

Опис поліноміального ймовірнісного алгоритму для добування квадратного кореня за простим модулем можна знайти в [111, стор. 47–48], [3, стор. 215–217] або [75]. Першим такий алгоритм опублікував Тонеллі [147]. Ми притримувались викладу цього питання в [13, питання 2 до розділу V].

Звідність факторизації до добування квадратного кореня помічена в [130]. Співвідношення між задачами розпізнавання квадратичності і добування кореня з одного боку, та факторизацією з іншого, переносяться із випадку $n = pq$ на випадок довільного складеного n. При цьому, як і у випадку $n = pq$, використовується в один бік Китайська теорема про остачі, а в інший — нескладне узагальнення твердження 4.4. Випадок модуля p^α зводиться до випадку простого модуля p (див. [13, §4 розділу V]).

Інші посилання на літературу з цих питань можна знайти в [64].

§ 5. Обчислення функції Ойлера

5.1. Випадок аргументу $n = pq$. Ми почнемо з аналізу задачі

Обчислення функції Ойлера від $n = pq$

Задано: $n = pq$, де $p \neq q$ непарні прості.

Обчислити: $\phi(n)$.

Слід зазначити, що співмножники p і q не входять в умову задачі. Однак якщо вони відомі, то $\phi(n)$ легко обчислюється за формулою $\phi(pq) = (p-1)(q-1)$. Таким чином обчислення функції Ойлера від аргументів виду $n = pq$ зводиться до факторизації таких чисел.

Має місце й обернене зведення. Припустимо, що для числа $n = pq$ відоме значення $\phi(n)$. Зауважимо, що $\phi(n) = n - p - q + 1$. Тому співмножники p і q визначаються із системи

$$\begin{cases} pq &= n \\ p+q &= n+1-\phi(n), \end{cases}$$

тобто є розв'язками квадратного рівняння

$$X^2 - X(n+1-\phi(n)) + n = 0.$$

5.2. Деякі узагальнення. Для $n = pq$ позначимо $\psi(n) = \text{НСК}(p-1, q-1)$. Очевидно, що значення $\phi(n)$ є кратним до $\psi(n)$. В попередньому пункті ми бачили, що n легко розкладається на співмножники у випадку, коли відомо значення $\phi(n)$. Виявляється, для факторизації числа n досить мати довільне число, кратне $\psi(n)$.

Точніше, факторизація чисел вигляду $n = pq$, які є добутком двох простих, зводиться за допомогою поліноміального ймовірнісного алгоритму до такої задачі.

Задано: $n = pq$ — добуток двох простих.

Знайти: m таке, що $\psi(n) \mid m$ і $m \leq 2^{(\log n)^c}$.

У другій умові на m, c служить позначенням для довільно фіксованої додатної константи. Таким чином, ця умова означає, що довжина запису m повинна обмежуватись деяким поліномом від довжини запису n.

Опис зведення, що пропонується нижче, супроводжується коментарями, які допоможуть зрозуміти, чому зведення влаштоване так, а не інакше. При аналізі зведення нам буде зручно на підставі наслідку II.2.13 утотожнювати мультиплікативну групу $\mathbb{Z}_n^*$ з ізоморфною їй групою $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$, а елемент $x \in \mathbb{Z}_n^*$ із парою $(y,z) \in \mathbb{Z}_p^* \times \mathbb{Z}_q^*$, де

$y = x \bmod p$ і $z = x \bmod q$. Нагадаємо, що x отримується з (y,z) ефективним чином, як і навпаки.

Перейдемо до опису зведення, тобто деякого алгоритму факторизації чисел $n = pq$, який має змогу отримувати від оракула певну підказку у вигляді числа m із зазначеними вище властивостями.

Вхід: $n = pq$, де $p \neq q$ непарні прості.

- Отримати від оракула значення m.
  Коментар. Оскільки $(p-1) \mid m$ і $(q-1) \mid m$, то для кожного $x \in \mathbb{Z}_n^*$ за малою теоремою Ферма $x^m \equiv 1 \pmod p$ і $x^m \equiv 1 \pmod q$, звідки
  $$x^m \equiv 1 \pmod n. \tag{1}$$
  Якщо (1) виконується для всіх $x \in \mathbb{Z}_n^*$, то m має бути парним. Щоб пересвідчитись у цьому, досить взяти $x = -1$.
- Зменшити значення m вдвічі.
- Незалежним чином вибрати k випадкових елементів $x_1, \ldots, x_k \in \mathbb{Z}_n^*$ і, використовуючи бінарний метод, піднести їх до степеня m за модулем n.
- Якщо для всіх $x = x_i$ з $i \leq k$ справджується конгруенція (1), то ще раз виконати два попередні пункти. Якщо хоча б в одному випадку (1) порушується, то перейти до наступного пункту.
  Коментар. Перед виконанням наступного пункту маємо значення m, для якого (1) виконується не для всіх $x \in \mathbb{Z}_n^*$. Крім того, з імовірністю щонайменше $1 - 2^{-k}$ для всіх $x \in \mathbb{Z}_n^*$ маємо
  $$x^{2m} \equiv 1 \pmod n \tag{2}$$
  (де ймовірність вимірюється за всіма випадковими виборами, що робилися на попередніх кроках, і від яких власне й залежить значення параметру m в поточний момент роботи алгоритму). Справді, множина елементів x, для яких виконується конгруенція (2), утворює підгрупу в $\mathbb{Z}_n^*$. Якщо ця підгрупа власна, то з теореми Лагранжа випливає, що вона містить не більш як половину всіх елементів із $\mathbb{Z}_n^*$. Тому ймовірність того, що всі k випадкових елементів $x_1, \ldots, x_k$ виявились із цієї підгрупи, не перевищує 2^{-k}.
- За допомогою алгоритму Евкліда обчислити значення $\text{НСД}(x_1^m - 1, n), \ldots, \text{НСД}(x_k^m - 1, n)$ і перше з них, не рівне ні 1, ні n, подати на вихід.

Оцінимо, з якою ймовірністю описаний алгоритм подає на вихід якийсь із нетривіальних дільників p або q числа n. Як зазначалось, із

ймовірністю принаймні $1-2^{-k}$ для останнього значення параметру m порівняння (2) має місце для всіх $x \in \mathbb{Z}_n^*$ в той час, коли порівняння (1) виконується не для всіх $x \in \mathbb{Z}_n^*$. Припустимо, що обидва ці факти справді мають місце. Звідси робимо такі висновки. По-перше, m не ділиться на $\phi(n) = (p-1)(q-1)$. По-друге, для кожного $x \in \mathbb{Z}_n^*$ степінь x^m є квадратним коренем з 1 в $\mathbb{Z}_n^*$. Далі розглянемо три випадки.

Випадок 1: $(p-1) \mid m$, $(q-1) \nmid m$.

Для $x = (y, z)$ маємо $x^m = (y^m, z^m)$, де піднесення до степеня відбувається в групах $\mathbb{Z}_n^*$, $\mathbb{Z}_p^*$ та $\mathbb{Z}_q^*$, відповідно. Для всіх y в $\mathbb{Z}_p^*$ маємо за малою теоремою Ферма $y^m = 1$. Оскільки для всіх x в $\mathbb{Z}_n^*$ виконується $x^{2m} = 1$, то маємо $z^{2m} = 1$ для всіх z в $\mathbb{Z}_q^*$, зокрема для z, що є твірним елементом групи $\mathbb{Z}_q^*$. Звідси отримуємо, що $2m = d(q-1)$ для деякого цілого d, і $m = d\frac{q-1}{2}$. Оскільки $x^m \neq 1$ для деяких x, то $z^m = -1$ для деяких z. Тому d повинно бути непарним. Використовуючи критерій Ойлера, отримуємо $z^m = \left(\frac{z}{q}\right)^d = \left(\frac{z}{q}\right)$ для всіх z. Відтак рівно для половини z із $\mathbb{Z}_q^*$ маємо $z^m = 1$, а для іншої половини $z^m = -1$ (див. пункт **b** вправи 1.9). Таким чином, для випадкового $x = (y, z)$ з ймовірністю $1/2$ виконується $x^m = (1, 1)$, і з такою ж ймовірністю $x^m = (1, -1)$. Іншими словами, завжди $x^m \equiv 1 \pmod{p}$, але з ймовірністю $1/2$ для випадкового x маємо $x^m \equiv -1 \pmod{q}$.

Випадок 2: $(q-1) \mid m$, $(p-1) \nmid m$.

Цей випадок симетричний до попереднього. Ті ж міркування показують, що завжди $x^m \equiv 1 \pmod{q}$, але у половині випадків $x^m \equiv -1 \pmod{p}$.

Випадок 3: $(p-1) \nmid m$, $(q-1) \nmid m$.

Такими ж міркуваннями, як у випадку 1, встановлюємо, що для випадкового $(y, z) \in \mathbb{Z}_p^* \times \mathbb{Z}_q^*$ степінь (y^m, z^m) з однаковою ймовірністю $1/4$ рівний одному з елементів $(1,1)$, $(1,-1)$, $(-1,1)$, $(-1,-1)$. Отже, для випадкового $x \in \mathbb{Z}_n^*$ з ймовірністю $1/2$ виконуються конгруенції $x^m \equiv 1$ за одним із модулів p або q та $x^m \equiv -1$ за іншим із цих модулів.

Як бачимо, в усіх випадках з імовірністю $1/2$ для випадкового x число $x^m - 1$ ділиться рівно на одне із чисел p або q, тобто НСД $(x^m - 1, n)$ є нетривіальним дільником числа $n = pq$. Ймовірність того, що цього не трапиться для жодного з k випадкових $x_1, \ldots, x_k$, дорівнює 2^{-k}.

У підсумку, ймовірність того, що алгоритм не видасть нетривіального дільника числа n, не перевищує $2^{-k} + 2^{-k}$, де один доданок береться із щойно отриманої оцінки, а другий обмежує ймовірність того, що порушується зроблене на початку наших міркувань припущення про параметр m. Таким чином, алгоритм досягає мети з імовірністю щонайменше $1 - 2^{-k+1}$.

ВПРАВИ

5.1. Число 9991 є добутком двох простих і $\phi(9991) = 9792$. Розкласти його на множники.

5.2. Прослідкувати роботу алгоритму із пункту 5.2 на вході $n = 35$, припустивши, що оракул робить підказку $m = 48$.

ЛІТЕРАТУРА

Зробимо кілька зауважень про зв'язок між задачами обчислення функції Ойлера від довільного аргументу та факторизації. Обчислення $\phi(n)$ в загальному випадку зводиться до факторизації аргументу n безпосередньо за формулою для функції Ойлера, встановленою в пункті II.2.4. Існування оберненої звідності доведено за РГР в [45]. У [64] зауважується, що із [45] випливає ймовірнісна звідність факторизації до обчислення функції Ойлера. Зокрема, з [45] виводиться звідність, представлена нами в пункті 5.2 (див. також [111, §2.2 розділу IV]).

§ 6. Первісні корені за простим модулем

6.1. Розпізнавання. Тепер розглянемо задачу

РОЗПІЗНАВАННЯ ПЕРВІСНОГО КОРЕНЯ ЗА ПРОСТИМ МОДУЛЕМ

Задано: $p, g \in \mathbb{N}$, де p просте.

Розпізнати: чи $g \bmod p$ породжує $\mathbb{Z}_p^*$?

Для цієї задачі невідомо ні детермінованих, ані ймовірнісних поліноміальних алгоритмів. Ситуація змінюється, якщо задано розклад на прості множники числа $p - 1 = q_1^{\alpha_1} \ldots q_s^{\alpha_s}$. В цьому разі можна застосувати пункт 2 твердження 1.2, згідно з яким g є первісним коренем за модулем p тоді і лише тоді, коли $g^{(p-1)/q_i} \not\equiv 1 \pmod{p}$ для всіх $i \leq s$. Останні умови перевіряються ефективно, адже ліва частина легко обчислюється за модулем p за допомогою бінарного методу піднесення до степеня.

Зауважимо, що тим самим встановлено поліноміальну звідність розпізнавання первісного кореня за простим модулем до факторизації.

6.2. Породження. Далі займемося задачею

ЗНАХОДЖЕННЯ ПЕРВІСНОГО КОРЕНЯ ЗА ПРОСТИМ МОДУЛЕМ

Задано: просте p.

Знайти: $g \in \mathbb{N}$ таке, що $g \bmod p$ породжує $\mathbb{Z}_p^*$.

Як і для розпізнавання, для знаходження первісного кореня ефективних алгоритмів невідомо. Є ймовірнісне зведення задачі знаходження до розпізнавання, яке забезпечується таким простим Лас Вегас алгоритмом.

Вибираємо в $\mathbb{Z}_p^*$ випадковий елемент g і перевіряємо, чи він є первісним коренем. Якщо так, то подаємо його на вихід; якщо ні, то звітуємо про невдачу. За пунктом 1 твердження 1.2 в $\mathbb{Z}_p^*$ є рівно $\phi(p-1)$ первісних коренів. Отже, ймовірність невдачі цього алгоритму дорівнює $1 - \frac{\phi(p-1)}{p-1}$. За твердженням II.2.15 це не більше, ніж $1 - 1/(6 \ln\ln(p-1))$. t-кратне повторення алгоритму зробить ймовірність невдачі не більшою від $(1 - 1/(6 \ln\ln(p-1)))^t$, що за за нерівністю B.4 не перевищує $\exp(-\frac{t}{6 \ln\ln(p-1)})$. Взявши $t = \lceil 6k \ln\ln(p-1) \rceil$, забезпечимо ймовірність невдачі, нижчу від e^{-k}.

В поєднанні з результатом попереднього пункту, щойно описане зведення дає поліноміальний Лас Вегас алгоритм для такої послабленої версії задачі знаходження первісного кореня.

Задано: просте p і розклад $p - 1 = q_1^{\alpha_1} \dots q_s^{\alpha_s}$.

Знайти: g — первісний корінь за модулем p.

Сформулюємо ще один варіант задачі знаходження первісного кореня, для якого відомо поліноміальний ймовірнісний алгоритм.

Задано: $n \in \mathbb{N}$.

Знайти: p і g, де p просте, $n \le p < 2n$,
g — первісний корінь за модулем p.

Для розв'язання цієї задачі спершу застосовується запропонована в [70] поліноміальна ймовірнісна процедура породження випадкового числа x із проміжку від n до $2n$ разом із його розкладом на прості співмножники $x = q_1^{\alpha_1} \dots q_s^{\alpha_s}$. Далі тестується на простоту число $p = x + 1$; якщо тест не пройдено, то вибирається інше x. В результаті у заданому проміжку знайдено просте p з відомим розкладом числа $p-1$, для якого первісний корінь g шукається як описано вище.

Зауважимо, що описаний алгоритм породжує пару p і g, де p — випадкове просте число між n і $2n$.

ВПРАВИ

6.1. Показати, що задача знаходження *всіх* первісних коренів за заданим простим модулем p ефективно зводиться до задачі знаходження *хоча б одного* первісного кореня.

6.2. ([13]) Довести, що 3 є первісним коренем будь-якого простого числа Ферма, більшого від 3, тобто простого числа вигляду $2^m + 1$, де $m > 1$.

ЛІТЕРАТУРА

Якщо справедлива РГР, то найменший первісний корінь за модулем p має величину $O(\log^6 p)$. Зрозуміло, що в цьому випадку відшукання первісного кореня зводиться до його розпізнавання за поліноміальний час детерміновано. Іншим наслідком є звідність знаходження первісного кореня за модулем p до задачі логарифмування за тим же модулем, яка розглядається в наступному параграфі. Деталі в [64].

§ 7. Дискретний логарифм

7.1. Складність дискретного логарифмування. Якщо $g^y \equiv x \pmod p$, де g — первісний корінь за простим модулем p і $p \nmid x$, то y називається *дискретним логарифмом* або *індексом* числа x за основою g і модулем p. Пишуть $y = \operatorname{ind}(x, g, p)$ або $y = \operatorname{ind}_g x$, якщо зрозуміло, про який модуль p йдеться. Для однозначності можна вважати, що $0 \le \operatorname{ind}_g x < p - 1$.

ПРИКЛАД 7.1. Озираючись на приклад 1.1, бачимо, що $\operatorname{ind}(1,5,23) = 0$, $\operatorname{ind}(2,5,23) = 2$, $\operatorname{ind}(3,5,23) = 16$, $\operatorname{ind}(4,5,23) = 4$, $\operatorname{ind}(5,5,23) = 1$, $\operatorname{ind}(6,5,23) = 18$.

З поняттям дискретного логарифму пов'язана наступна задача, на важкості якої грунтується надійність чималої кількості криптосистем.

ДИСКРЕТНЕ ЛОГАРИФМУВАННЯ ЗА ПРОСТИМ МОДУЛЕМ

Задано: $x, g, p \in \mathbb{N}$, де p просте, $p \nmid x$,
g — первісний корінь за модулем p.

Знайти: y таке, що $g^y \equiv x \pmod p$.

Слід констатувати, що за станом наших сьогоднішніх знань дискретне логарифмування є важкою задачею. Час найкращих алгоритмів для логарифмування має таку ж асимптотику як і для факторизації. Ця асимптотика виражається формулами (1) і (2) з § 3, причому остання стосується евристичних оцінок. Проте про звідності між задачами факторизації та логарифмування нічого невідомо.

7.2. Алгоритм Сільвера-Поліга-Гелмана. Для криптоаналізу важливо знати ті часткові випадки, коли дискретне логарифмування можна провести ефективно. Таким є випадок, що $p-1$ має лише невеликі прості дільники. Точніше, натуральне число назвемо *s-гладким*, якщо жоден його простий дільник не перевищує s. У подальшому для модуля p через s будемо позначати найбільший простий дільник числа $p-1$. Таким чином, $p-1$ є s-гладким. Є алгоритм для знаходження $\mathrm{ind}(x,g,p)$, час роботи якого обмежений поліномом від $\max\{s, \log p\}$.

Алгоритм Сільвера-Поліга-Гелмана

Вхід: x, g, $p \in \mathbb{N}$, де p просте, $p \nmid x$, g первісний корінь за модулем p.

Вихід: $y = \mathrm{ind}(x,g,p)$.

Робота алгоритму відбувається так. Спочатку факторизуємо число $p-1$ безпосереднім діленням його на числа, що не перевищують s. Це займе не більше від $s \log p$ операцій ділення. Нехай $p-1 = \prod_{q|(p-1)} q^{\alpha}$ — розклад $p-1$ на прості множники. Для кожного члена розкладу q ми збираємось знайти лишок $y \bmod q^{\alpha}$. Після того, як це буде зроблено, y легко реконструюється за Китайською теоремою про остачі, точніше, за допомогою алгоритму, що випливає з її доведення.

Отже, розглянемо число q, яке входить у розклад числа $p-1$ в степені α. Запишемо

$$y \equiv \sum_{i=0}^{\alpha-1} y_i q^i \pmod{q^{\alpha}}, \text{ де } 0 \le y_i \le q-1.$$

Нашим завданням є визначення послідовності $y_0, \ldots, y_{\alpha-1}$. Зауважимо, що

$$\left(y - (y \bmod q^i)\right) \frac{p-1}{q^{i+1}} \equiv \left(\sum_{k=i}^{\alpha-1} y_k q^k\right) \frac{p-1}{q^{i+1}} \equiv y_i \frac{p-1}{q} (\mathrm{mod} p-1)$$

для $i = 0, 1, \ldots, \alpha-1$. Звідси, з використанням малої теореми Ферма для модуля p і числа g, маємо

$$g^{\left(y-(y \bmod q^i)\right)\frac{p-1}{q^{i+1}}} \equiv g^{y_i \frac{p-1}{q}} \pmod p,$$

що переписуємо як

$$\left(x/g^{y \bmod q^i}\right)^{\frac{p-1}{q^{i+1}}} \equiv \left(g^{\frac{p-1}{q}}\right)^{y_i} \pmod p. \qquad (1)$$

Ділення в основі лівої частини є операцією в $\mathbb{Z}_p^*$.

Покажемо, як з останньої конгруенції визначити y_i. При $i = 0$ отримуємо умову

$$x^{\frac{p-1}{q}} \equiv \left(g^{\frac{p-1}{q}}\right)^{y_0} \pmod p. \qquad (2)$$

За допомогою бінарного методу піднесення до степеня обчислюємо послідовність чисел

$$\left(g^{\frac{p-1}{q}}\right)^j \text{ для } j = 0, 1, \ldots, q-1 \qquad (3)$$

(яка буде потрібна і для всіх інших $i > 0$). Зауважимо, що всі елементи послідовності різні, бо g — первісний корінь. Обчислюємо також степінь $x^{(p-1)/q} \bmod p$, який з огляду на (2) мусить бути рівним одному із членів послідовності (3). Коефіцієнт y_0 дорівнює відповідному показнику j.

Припустимо, що вже знайдено $y_0, \ldots, y_{i-1}$. Тоді ми в стані обчислити ліву частину конгруенції (1) за модулем p, оскільки $y \bmod q^i = \sum_{k=0}^{i-1} y_k q^k$. Порівнюючи результат з елементами послідовності (3), знаходимо y_i.

Це завершує опис алгоритму.

ВПРАВИ

7.1. **a)** Довести очікуване для логарифму співвідношення: якщо g і g' — два первісні корені за простим модулем p, і число x не ділиться на p, то $\mathrm{ind}_g x = \mathrm{ind}_g g' \, \mathrm{ind}_{g'} x$.

b) Показати, що логарифмування за *всіма* можливими основами зводиться до логарифмування за будь-якою *одною* основою.

7.2. За допомогою алгоритму Сільвера-Поліга-Гелмана обчислити **a)** $\mathrm{ind}(11,2,13)$; **b)** $\mathrm{ind}(25,2,37)$; **c)** $\mathrm{ind}(6,5,97)$.

ЛІТЕРАТУРА

Огляд відомих алгоритмів дискретного логарифмування зроблено в [64, 114, 111]. Алгоритм Сільвера-Поліга-Гелмана описано в [126] (див. також [111, стор. 98], [42, стор. 17] або [3, стор. 217]).

Поняття логарифму можна розглядати для будь-якої алгебраїчної структури із множенням. Про стан справ із логарифмуванням у скінченних полях дивись [64] і цитовану там літературу. Там же згадане ймовірнісне зведення факторизації до дискретного логарифмування, поширеного на будь-який натуральний модуль. Логарифмування за модулем p^{α} зводиться до логарифмування за простим модулем p (див. [111, стор. 103]).

§ 8. Підсумок

Підведемо риску під нашим дослідженням складності задач цілочисельної арифметики.

- Є ефективні тести, за допомогою яких легко розпізнати, простим чи складеним є задане число. Як наслідок, для практичних потреб легко вибрати просте число заданого порядку.
- Є ефективний алгоритм для добування квадратних коренів за простим модулем p. У випадках $p = 4m + 3$ і $p = 8m + 5$ алгоритм працює детермінованим чином і є особливо простим.
- Невідомо ефективних алгоритмів для факторизації і дискретного логарифмування — ці задачі на сьогодні є важкими.
- Є ефективне зведення добування квадратних коренів за модулем $n = pq$ до факторизації числа $n = pq$, і навпаки — факторизації до добування кореня.
- Еквівалентними задачами є також обчислення функції Ойлера $\phi(n)$ від $n = pq$ і факторизація числа $n = pq$. Більше того, щоб факторизувати $n = pq$, досить мати довільне кратне числа $\psi(n) = \text{НСК}\,(p-1, q-1)$.
- Задача знаходження первісного кореня за простим модулем p зводиться до задачі розпізнавання, яка у свою чергу зводиться до факторизації числа $p - 1$.

Розділ V.

Криптосистеми з відкритим ключем

§ 1. Концепція

1976 рік відкрив сучасний етап у криптографії. Для новітньої криптографії характерною рисою є поява принципово нових криптографічних задач, а також принципово нових розв'язків задач класичних. Тому часто говорять про революцію у галузі криптографії. Поступ, що відбувся у 1976 році, пов'язаний з іменами американських математиків Вайтфілда Діффі та Мартіна Гелмана, а також Ральфа Меркле, які розвинули ідеологію відкритого ключа.

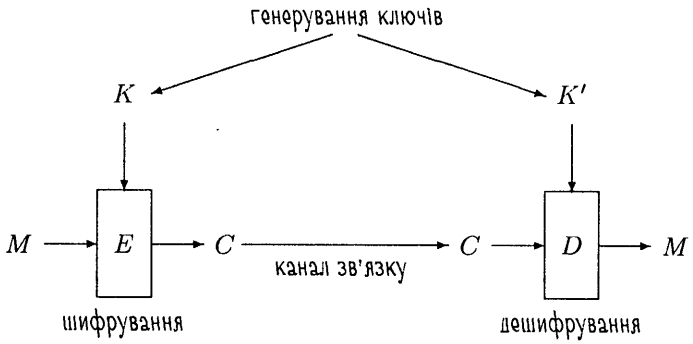
У криптосистемі з відкритим ключем процедура шифрування є загальнодоступною. Це однак не означає як у традиційних криптосистемах, що загальнодоступним є також дешифрування. Зупинимось докладніше на цій парадоксальній з першого погляду тезі.

Пригадаємо нашу базову криптографічну схему з пункту I.1.1. Вона передбачає поняття ключа K, який використовується як при шифруванні, так і при дешифруванні. При розгляді у § II.3 афінних шифрів ми дещо модернізували нашу термінологію. Безпосередньо за допомогою ключа K здійснювалось лише шифрування, а для дешифрування використовувався *дешифруючий ключ* K'. Оскільки $K' = K'(K)$ легко отримувався з K, то такий поділ ключа був не принциповим відходом від початкової схеми, а лише справою домовленості та зручності. В такому контексті ключ K природно назвати *шифруючим*. Для повідомлення M і криптотексту C матимемо співвідношення $C = E_K(M)$ і

$M = D_{K'}(M)$, де K і K' — шифруючий і дешифруючий ключі, а E і D — алгоритми шифрування та дешифрування, відповідно.

У всіх криптосистемах, які ми розглядали досі, подібно до згаданого афінного шифру дешифруючий ключ знаходився за шифруючим ефективно. Раніше ми (як і все криптографічне співтовариство до 1976 року) й не замислювались, що такий стан речей можна змінити, і то з неабиякою користю. А якраз в допущенні того, що знаходження K' за K може бути важким, і полягає ідея, яка визначила подальший напрям розвитку криптографії.

Поняття *криптосистеми з відкритим ключем* включає такі об'єкти (див. малюнок 1):



Малюнок 1. Нова криптографічна схема.

- Алфавіт $\mathcal{A}$, в якому записуються повідомлення (відкриті тексти), і алфавіт $\mathcal{B}$, в якому записуються криптотексти.
- Простір ключів $\mathcal{K}$ (множина слів у деякому алфавіті).
- Алгоритм *генерування ключів*. Це поліноміальний ймовірнісний алгоритм, який на вході k, де $k \in \mathbb{N}$ — записаний в унарній системі *параметр надійності*, видає випадкову пару K, $K' \in \mathcal{K}$. K називається *відкритим ключем* і використовується для шифрування, а K' називається *таємним ключем* і використовується для дешифрування.
- Поліноміальний детермінований алгоритм шифрування E, який отримує на вхід повідомлення M і відкритий ключ K, а видає криптотекст C, що записуємо як $C = E_K(M)$.

- Поліноміальний детермінований алгоритм дешифрування D, який отримує на вхід криптотекст C і таємний ключ K', а видає відкритий текст C, що записуємо як $M = D_{K'}(C)$.

Перелічені алгоритми мають задовольняти такі умови:

- Якщо пара (K, K') породжена алгоритмом генерування ключів, то з $C = E_K(M)$ випливає $M = D_{K'}(C)$ для будь-якого відкритого тексту M.
- Немає (чи, принаймні, невідомо) жодного ефективного алгоритму, який за відомими $C = E_K(M)$ і K знаходив би M.

Остання умова забезпечує надійність криптосистеми навіть тоді, коли із відкритого ключа K не робиться таємниці. З цієї умови випливає зокрема, що немає ефективного алгоритму знаходження за K такого K', що пара (K, K') могла б бути породженою алгоритмом генерування ключів.

Зауважимо, що коли відкритий ключ є доступним для суперника, останній має достеменно ті ж можливості криптоаналізу, які для традиційних криптосистем називалися атакою з вибраним відкритим текстом. Як і раніше, сильнішим видом криптоаналізу залишається атака з вибраним криптотекстом.

Криптосистеми з відкритим ключем ще називають *асиметричними*. В цьому контексті класичні криптосистеми називають *симетричними*.

Тепер з'ясуємо переваги нової криптографічної схеми над старою. Уявімо комунікаційну мережу, якою користуються n абонентів — Аліса, Боб, Вітольд, Габріеля,... Припустимо, що кожна пара абонентів може мати свої маленькі таємниці і хоче мати можливість спілкуватися таємно від решти абонентів. Оскільки всіх пар є $n(n-1)/2$, то саме таку кількість ключів ми потребуємо, коли послуговуємося старою симетричною схемою. Нова асиметрична схема дозволяє влаштувати довірливе спілкування в мережі оптимальнішим чином. Кожен абонент, власноручно або через менеджера мережі, генерує власну пару ключів (K, K'). Аліса стає власником пари ключів (K_A, K'_A), Боб — (K_B, K'_B) і т.д. Кожен абонент свій приватний ключ зберігає в таємниці, в той час як список всіх відкритих ключів $\langle K_A, K_B, K_B, K_\Gamma, \ldots \rangle$ є у загальному доступі. Коли Боб хоче послати Алісі повідомлення M, він посилає їй криптотекст $C = E_{K_A}(M)$. Оскільки тільки Аліса знає таємний ключ K'_A, лише вона здатна дешифрувати C. Таким чином, таємницю листування збережено з використанням лише n пар відкритого та таємного ключів, на противагу до необхідних раніше $n(n-1)/2$ ключів.

Наступні параграфи присвячені конкретним втіленням описаної вище схеми, а також її модифікаціям.

ЛІТЕРАТУРА

Піонерські роботи Діффі і Гелмана, та Меркле опубліковані в [20] та [119]. Цікавим вступом до предмету може служити ретроспективний огляд [19] (див. також [21]). У подальшому викладі ми не торкаємось класу криптосистем з відкритим ключем, пов'язаних із *задачею про рюкзак*. Про ці криптосистеми та успіхи в їх криптоаналізі йдеться в [10].

§ 2. RSA

2.1. Опис системи. Запропонована 1977 року система RSA є чи не найпопулярнішою криптосистемою з відкритим ключем. Назва системи утворена з перших літер імен її винахідників — Рональда Райвеста, Аді Шаміра та Леонарда Адлемана.

Генерування ключів. Вибирають два досить великі прості числа p і q. Для їх добутку $n = pq$ значення функції Ойлера дорівнює $\phi(n) = (p-1)(q-1) = n-p-q+1$. Далі випадковим чином вибирають елемент e, що не перевищує значення $\phi(n)$ і взаємно простий з ним. Іншими словами, e є випадковим елементом із множини $\mathbb{Z}^*_{\phi(n)}$. Для e за алгоритмом Евкліда знаходить елемент d, обернений до e в $\mathbb{Z}^*_{\phi(n)}$, тобто такий, що $d < \phi(n)$ і

$$ed \equiv 1 \,(\mathrm{mod}\, \phi(n)). \tag{1}$$

Як результат покладають:

Відкритий ключ: e, n.

Таємний ключ: d.

Шифрування відбувається блоками. Для цього повідомлення записують у цифровій формі і розбивають на блоки так, що кожен блок позначає число, яке не перевищує n. Скажімо, якщо блок записаний у вигляді двійкового слова довжини m, то повинна виконуватись нерівність $2^m < n$. Блок M розглядається як елемент кільця $\mathbb{Z}_n$ і як такий, може підноситись до степеня за модулем n.

Алгоритм шифрування E у системі RSA полягає у піднесенні M до степеня e. Записуємо це так:

$$E(M) = M^e \bmod n.$$

В результаті отримується блок криптотексту $C = E(M)$, який також є цифровим записом якогось елемента кільця $\mathbb{Z}_n$.

Дешифрування. Алгоритм дешифрування D блоку криптотексту C полягає у піднесенні C до степеня d, тобто

$$D(C) = C^d \bmod n.$$

Приклад 2.1. Нехай $p = 53$ і $q = 67$. Тоді $n = 3551$ і $\phi(n) = 3432$. Візьмемо $e = 1021$ — за допомогою розширеного алгоритму Евкліда легко перевірити, що НСД $(1021, 3432) = 1$ (див. § II.2). Одночасно обчислюємо $d = 1021^{-1} \bmod 3432 = 1237$. Ключі вибрано.

Відкритий ключ $e = 1021$ і $n = 3551$ оприлюднюємо. Тепер будь-хто може послати нам зашифроване повідомлення. Припустимо, один із ділових партнерів вирішив послати нам вказівку ПРОДАЙ. Спочатку він перетворює своє повідомлення у цифрову форму, замінюючи кожну літеру її двоцифровим десятковим номером в алфавіті: 1920 1805 0013 (нагадаємо домовленість починати нумерацію з нуля). Видно, що з нашим модулем n цифрове повідомлення варто розбивати на блоки по 4 цифри, як це і зроблено. При шифруванні перший блок 1920 перетворюється у $1920^{1021} \bmod 3551 = 2393$. Піднесення до степеня ефективно виконується за допомогою бінарного методу, описаного в пункті III.2.2. Таким же чином шифруються наступні два блоки, і в результаті виходить криптотекст 2393 1788 2188.

Отримавши цей криптотекст, проводимо дешифрування піднесенням кожного блоку до степеня $d = 1237$ за модулем $n = 3551$. Зацікавлений читач може власноручно переконатись, що $2393^{1237} \bmod 3551 = 1920$ і т.д.

2.2. Коректність. Слід пересвідчитись, що $D(E(M)) = M$ для будь-якого повідомлення M. Сформулюємо це у вигляді

Твердження 2.2. *Нехай $n = pq$ є добутком двох різних простих чисел. Якщо $ed \equiv 1 \,(\mathrm{mod}\, \phi(n))$, то для всіх $x \in \mathbb{Z}_n$*

$$x^{ed} \equiv x \,(\mathrm{mod}\, n).$$

В пункті IV.5.2 ми ввели функцію ψ таку, що $\psi(n) = \mathrm{НСК}\,(p-1, q-1)$ для $n = pq$. Ми отримаємо твердження 2.2 як наслідок більш загального

Твердження 2.3. *Нехай $n = pq$ є добутком двох різних простих чисел. Співвідношення*

$$x^t \equiv x \,(\mathrm{mod}\, n) \tag{1}$$

виконується для всіх цілих x тоді і лише тоді, коли

$$t \equiv 1 \,(\mathrm{mod}\, \psi(n)). \tag{2}$$

Доведення. Оскільки $n = pq$, то умова (1) рівносильна одночасному виконанню двох умов

$$x^t \equiv x \,(\mathrm{mod}\, p) \quad \text{та} \tag{3}$$

$$x^t \equiv x \,(\mathrm{mod}\, q). \tag{4}$$

Розглянемо першу з цих умов.

Якщо $p \mid x$, то конгруенція (3) справедлива незалежно від того, яким є t. Отже, виконання (3) для всіх x еквівалентне виконанню конгруенції

$$x^{t-1} \equiv 1 \,(\mathrm{mod}\, p) \tag{5}$$

для всіх $x \in \mathbb{Z}_p^*$. Якщо $(p-1) \mid (t-1)$, то (5) має місце за малою теоремою Ферма. Умова $(p-1) \mid (t-1)$ є також необхідною для виконання (5) для всіх $x \in \mathbb{Z}_p^*$, оскільки в $\mathbb{Z}_p^*$ є елементи порядку якраз $p-1$ (твердження Б.6). Таким чином, справедливість конгруенції (3) для всіх x рівносильна подільності $t-1$ на $p-1$.

Для (4) справедливий такий же факт з q замість p. Отже, одночасне виконання умов (3) і (4) еквівалентне подільності числа $t-1$ на НСК $(p-1, q-1) = \psi(n)$, тобто конгруенції (2). Що й слід було довести. ∎

Щоб вивести твердження 2.2 із твердження 2.3, покладемо $t = ed$. Умова (2) випливає з умови (1), бо $\psi(n) \mid \phi(n)$.

2.3. Ефективність. Алгоритм генерування ключів використовує процедуру породження простих чисел (пункт IV.2.7) і розширений алгоритм Евкліда для обчислення НСД $(e, \phi(n))$ і $d = e^{-1} \bmod \phi(n)$ (§ II.2). В алгоритмах шифрування та дешифрування піднесення до степеня виконується за допомогою бінарного методу (пункт III.2.2).

2.4. Надійність. Щоб криптосистема вважалась надійною, необхідно щоб задача визначення повідомлення за криптотекстом і відкритим ключем була важкою. В нашому випадку останню задачу можна сформулювати так.

Розкриття RSA

Задано: e, n, y, де $n = pq$ і НСД $(e, \phi(n)) = 1$.

Знайти: x таке, що $x^e \equiv y \,(\mathrm{mod}\, n)$.

Як бачимо, задача розкриття RSA є дослівно задачею добування із заданого цілого числа кореня e-го степеня за модулем $n = pq$ (при перелічених умовах). На сьогоднішній момент для цієї задачі невідомо

ніякого ефективного алгоритму. Однак не доведено, що такого алгоритму не існує. На жаль, така ситуація є типовою для всіх криптосистем з відкритим ключем, що мають практичний інтерес. Єдине, що ми можемо зробити, і що є нашим завданням у цьому пункті, це розглянути деякі специфічні методи криптоаналізу для RSA і оцінити їхню (без)перспективність.

Будь-яку асиметричну криптосистему можна зламати, вказавши ефективний спосіб визначення таємного ключа за відкритим. У нашому випадку це означає, що розкриття RSA зводиться до задачі

Знаходження таємного ключа для RSA

Задано: e, n, де $n = pq$ і НСД $(e, \phi(n)) = 1$.

Знайти: d таке, що $x^{ed} \equiv x \,(\mathrm{mod}\, n)$ для всіх x.

З алгоритму генерування ключів безпосередньо випливає, що остання задача зводиться до обчислення значення функції Ойлера $\phi(n)$. Як нам відомо з пункту IV.5.1, обчислення функції Ойлера від аргументів такого типу є задачею еквівалентною до знаходження співмножників p і q числа n. Отже, спроба факторизувати модуль $n = pq$ є найочевиднішим шляхом до розкриття RSA. На щастя, як зазначалось в § IV.3, на сьогодні це є безнадійною справою для n порядку 10^{200}. Тому при генеруванні ключа, p і q рекомендується вибирати із приблизно сотнею десяткових цифр кожне. Вибір таких чисел повинен бути справді випадковим, щоб уникнути можливої факторизації якимось із вузькоспеціальних методів (див. обговорення в пункті IV.2.7 і § IV.3 разом із задачами). Звідси також високі вимоги до якості генератора псевдовипадкових бітів, що використовується алгоритмом породження простих чисел.

Природно виникає питання, чи не можна розв'язати задачу визначення таємного ключа, обходячись без факторизації. За твердженням 2.3, цю задачу можна переформулювати як задачу знаходження такого d, що $ed-1$ ділиться на $\psi(n)$. Але для такого d число $m = ed-1$ можна використати для розкладу n на множники за допомогою ймовірнісної процедури, описаної в пункті IV.5.2. Отже, визначення таємного ключа для RSA є таким же важким, як факторизація модуля n.

Підсумовуючи наші аргументи, отримуємо

Твердження 2.4. *Знаходження таємного ключа для RSA є еквівалентним до факторизації чисел, що є добутком двох простих, відносно поліноміальної ймовірнісної звідності.*

Зауважимо, що для деяких повідомлень M має місце рівність $E(M) = M$. Числові еквіваленти таких повідомлень є розв'язками конгруенції $x^e \equiv x \pmod n$ в $\mathbb{Z}_n$, кількість яких нескладно порахувати (див. вправу 2.5).

Виявляється, що доля повідомлень, які подібно до наведеного прикладу можуть бути розкриті якимось простим специфічним способом, не може бути великою, хіба що *всі* повідомлення можуть бути дешифровані подібним чином.

Щоб формалізувати цей факт, введемо такі позначення. Припустимо, що для кожної пари e і n таких, що НСД $(e, \phi(n)) = 1$, маємо деяку підмножину $Y_{e,n}$ кільця $\mathbb{Z}_n$. Через $\delta_n = \min_e \|Y_{e,n}\|/n$ позначимо мінімальну по e щільність $Y_{e,n}$ в $\mathbb{Z}_n$. Звуженням задачі розкриття RSA на родину підмножин $Y_{e,n}$ назвемо таку задачу.

Задано: e, n, y, де $n = pq$, НСД $(e, \phi(n))$ і $y \in Y_{e,n}$.
Знайти: x таке, що $x^e \equiv y \pmod n$.

Твердження 2.5. *Для довільної родини підмножин $Y_{e,n} \subseteq \mathbb{Z}_n$ задача розкриття RSA ймовірнісно зводиться до свого звуження на $Y_{e,n}$ за час, обмежений деяким поліномом від довжини входу і величини $1/\delta_n$.*

Таким чином, якщо не існує ймовірнісного поліноміального алгоритму для розкриття RSA, то нема й такого алгоритму, який би розкривав частку $1/\log^c n$ всіх можливих криптотекстів, де c — константа. Іншими словами, будь-який спосіб розкриття лише частки $1/\log^c n$ всіх можливих криптотекстів можна використати для розкриття абсолютно всіх криптотекстів.

Доведення. Припустимо, що в нас є оракул, який, отримавши запит $(e, n, \tilde y)$, видає деякий елемент $\tilde x \in \mathbb{Z}_n$ із властивістю $\tilde y \equiv \tilde x^e \pmod n$, справедливою завжди, коли $\tilde y \in Y_{e,n}$. Тоді RSA ламається таким ймовірнісним алгоритмом.

Вхід: e, n, y, де $n = pq$ і НСД $(e, \phi(n)) = 1$.

- Обчислити НСД (y, n). Якщо $1 <$ НСД $(y, n) < n$, то знайдено нетривіальний дільник модуля n, і криптосистема ламається визначенням таємного ключа d. Якщо НСД $(y, n) = n$, то подати на вихід $x = 0$. Якщо НСД $(y, n) = 1$, то продовжувати.
- Вибрати випадковий елемент $\hat x \in \mathbb{Z}_n$. Якщо НСД $(\hat x, n) > 1$, то теж отримано нетривіальний дільник модуля n. Якщо

$$\text{НСД}\,(\hat x, n) = 1, \tag{1}$$

то продовжувати.

- Обчислити

$$\tilde y = y\hat x^e \bmod n. \tag{2}$$

- Зробити оракулові запит $(e, n, \tilde y)$.
- Отримавши відповідь $\tilde x$, перевірити чи справді

$$\tilde y \equiv \tilde x^e \pmod n. \tag{3}$$

Якщо ні, визнати невдачу; інакше продовжувати.

- Обчислити і подати на вихід

$$x = \tilde x \hat x^{-1} \bmod n, \tag{4}$$

де $\hat x^{-1}$ — елемент, обернений до $\hat x$ в $\mathbb{Z}_n^*$.

Позначимо ймовірність невдачі описаного алгоритму через α і оцінимо її. У виродженому випадку, коли НСД $(y, n) > 1$, маємо $\alpha = 0$. Проаналізуємо роботу алгоритму на входах, для яких НСД $(y, n) = 1$.

Алгоритм зразу досягає успіху в тому щасливому випадку, коли НСД $(\hat x, n) > 1$. Це трапляється з імовірністю $1 - \phi(n)/n = 1/p + 1/q - 1/n$, тому першою нашою оцінкою є

$$\alpha \le \frac{\phi(n)}{n}. \tag{5}$$

Розглянемо випадок, коли має місце (1). Основою нашого аналізу є спостереження, що коли $\tilde y$ попадає в $Y_{e,n}$, то алгоритм справді видає відкритий текст x, що відповідає криптотекстові y, тобто виконується умова $y \equiv x^e \pmod n$. Справді, ця конгруенція випливає безпосередньо із співвідношень (2), (3) і (4).

Таким чином, алгоритм зазнає невдачі лише у випадку, коли $\hat x \in \mathbb{Z}_n^*$ і $\tilde y \notin Y_{e,n}$. Ймовірність першої з цих двох подій дорівнює $\phi(n)/n$. Оцінимо ймовірність другої події, припустивши, що перша має місце. Зробимо також припущення, що

$$\delta_n > 1 - \phi(n)/n. \tag{6}$$

Тоді для кожного e щільність множини $Y_{e,n} \cap \mathbb{Z}_n^*$ в $\mathbb{Z}_n^*$ не менша, ніж $\delta_n - (1 - \phi(n)/n)$. Щонайменше з такою ймовірністю $\tilde y$ потрапляє в $Y_{e,n}$, бо за умови (1) $\tilde y$ є випадковим елементом множини $\mathbb{Z}_n^*$. Підсумовуючи, отримуємо оцінку

$$\alpha \le \frac{\phi(n)}{n}\left(1 - (\delta_n - (1 - \phi(n)/n))\right) = \frac{\phi(n)}{n}\left(2 - \frac{\phi(n)}{n} - \delta_n\right).$$

Використовуючи нерівність $ab \le (\frac{a+b}{2})^2$, отримуємо звідси оцінку

$$\alpha \le (1 - \delta_n/2)^2. \tag{7}$$

Нагадаємо, що ми довели (7) на основі припущення (6). Якщо ж (6) не виконується, тобто $\phi(n)/n \le 1 - \delta_n$, то за (5) маємо нерівність $\alpha \le 1 - \delta_n$, звідки випливає та ж оцінка (7).

Щоб зменшити ймовірність невдачі, описаний алгоритм слід виконати k разів, як описано у пункті III.3.1. Для $k = \lceil t/\delta_n \rceil$ ймовірність невдачі буде меншою, ніж

$$(1 - \delta_n/2)^{2k} \le e^{-\delta_n k} \le e^{-t}.$$

■

ВПРАВИ

2.1. a) Сформувати відкритий і таємний ключі для системи RSA на основі $p = 47$, $q = 71$, $e = 19$.

b) Зашифрувати повідомлення КУПИ. При переході до цифрової форми замінювати кожну літеру її двоцифровим десятковим номером в алфавіті. Цифрове повідомлення розбивати на блоки по 4 цифри.

c) Дешифрувати криптотекст 3011 1211.

2.2. Підрахувати кількість повідомлень $M \in \mathbb{Z}_n$, для яких НСД $(M, n) > 1$. (Пересилання таких повідомлень слід виключити, бо з їх перехопленням суперник отримує нетривіальний дільник модуля n.)

2.3. Зламати RSA у випадку, коли

a) p і q — прості числа-близнята;

b) різниця $q - p$ невелика.

2.4. Аліса, Боб і Бернар є абонентами комунікаційної мережі. Для захисту інформації в мережі використовується система RSA, причому ключі генеруються і розподіляються між абонентами менеджером мережі.

a) ([142]) Боб і Бернар користуються відкритими ключами (e_1, n) та (e_2, n), де e_1 та e_2 взаємно прості. Аліса послала кожному з них одне і те ж повідомлення M. Пояснити, як можна визначити M за криптотекстами $C_1 = M^{e_1} \bmod n$ та $C_2 = M^{e_2} \bmod n$.

b) ([85]) Пояснити, як Боб може визначити Бернарів таємний ключ d_2, а Бернар — Бобів ключ d_1.

Обговорення подібної проблематики див. в [47].

2.5. a) Для $n = pq$, добутку двох різних простих, порахувати кількість елементів $x \in \mathbb{Z}_n$, які задовольняють конгруенцію $x^e \equiv x \pmod n$.

b) Знайти всі розв'язки конгруенції $x^3 \equiv x \pmod{15}$ в $\mathbb{Z}_{15}$.

2.6. Суперник перехопив криптотекст $C \in \mathbb{Z}_n$, зашифрований з допомогою відкритого ключа (e, n), і один за другим обчислює члени послідовності

$C^2 \bmod n, C^3 \bmod n, \ldots$. Довести, що зробивши не більш як $\psi(n)$ кроків, суперник гарантовано отримає відкритий текст. (Інша причина слідкувати, щоб $\psi(n)$ було великим чи, еквівалентно, НСД $(p-1, q-1)$ малим, відображена у вправі IV.3.2.)

2.7. Довести, що $\phi(\psi(n)) - 1$ ітерацій досить, щоб розкрити систему RSA методом ітерацій.

ЛІТЕРАТУРА

Оригінальною публікацією про систему RSA є [132]. Популярний виклад можна знайти в [16].

З метою пришвидшення процедури шифрування пропонувалось завжди використовувати експоненту $e = 3$ замість того, щоб щоразу вибирати її випадковим чином. Проте у [103] показано, що використання невеликого параметра e дає можливість супернику розкрити повідомлення у випадку, якщо воно розіслане кільком користувачам криптосистеми.

Твердження 2.5 наведене у [131] із посиланням на Яо. Воно показує, що система RSA однаково надійна (чи ненадійна) як для всіх повідомлень, так і для будь-якої малої їх частини. Таку однорідність теж можна вважати свідченням на користь надійності RSA. Інші властивості такого ж плану доведено в [101, 67, 73] (див. також вправу VI.1.4).

§ 3. Система Рабіна

Цю криптосистему запропоновано в [130].

Генерування ключів. Вибирають два великі прості числа p і q. Обчислюють їх добуток $n = pq$. Покладають

Відкритий ключ: n.

Таємний ключ: p, q.

Шифрування відбувається блоками подібно до системи RSA, згідно з формулою

$$E(M) = M^2 \bmod n.$$

Дешифрування. Якщо $E(M) = C$, то M є квадратним коренем числа C за модулем n. За умови НСД $(C, n) = 1$, в $\mathbb{Z}_n$ таких коренів є рівно чотири (пункт 3 твердження IV.4.1). Результати пунктів IV.4.3 і IV.4.2 дають ефективний алгоритм добування всіх квадратних коренів за модулем n, який використовує співмножники p і q (тобто таємний ключ!). Саме цей алгоритм використовується в системі Рабіна при дешифруванні. Після знаходження всіх чотирьох коренів з них вибирається той, який є числовим еквівалентом осмисленого тексту.

Зауваження 3.1. В системі Рабіна шифруюче відображення не є ін'єктивним, але може бути зробленим таким шляхом простої модифікації. Однозначності можна досягти за рахунок передачі разом із криптотекстом деякої додаткової незашифрованої інформації (вправа 3.2). Це справді необхідно, коли шифрується не текстова, а суто числова інформація.

Зауваження 3.2. При описі алгоритму шифрування ми виходили з припущення, що НСД $(C, n) = 1$ (це еквівалентно з НСД $(M, n) = 1$). Посилання повідомлень C, для яких НСД $(C, n) > 1$, слід виключити, бо з їх перехопленням суперник отримує нетривіальний дільник НСД (C, n) числа n, тобто дізнається таємний ключ.

Приклад 3.3. Нехай таємний ключ вибрано так: $p = 53$ і $q = 67$. Тоді відкритим ключем буде $n = 3551$.

Розглянемо шифрування повідомлення ПРОДАЙ. Як це було зроблено у прикладі 2.1, спочатку повідомлення записується у цифровій формі і розбивається на блоки по чотири цифри: 1920 1805 0013. Перший блок 1920 перетворюється у $1920^2 \bmod 3551 = 0462$. Подібно шифруються наступні два блоки, і в результаті виходить криптотекст 0462 1758 0169.

Припустимо тепер, що ми отримали криптотекст 1497. Для дешифрування слід з нього добути квадратні корені за модулем 3551. З цією метою добуваємо корені за простими модулями 53 і 67 із лишків $1497 \bmod 53 = 13$ і $1497 \bmod 67 = 23$, відповідно. Застосовуємо алгоритм із пункту IV.4.2. Модуль 53 належить до випадку 2, а 67 до випадку 1. Знаходимо $\sqrt{13} \bmod 53 = 15, 38$ і $\sqrt{23} \bmod 67 = 31, 36$. За допомогою алгоритму з Китайської теореми про остачі визначаємо чотири корені з 1497 за модулем 3551: $(15, 31) = 0969$, $(15, 36) = 1711$, $(38, 31) = 1840$, $(38, 36) = 2582$. Як зразу видно, лише другий корінь є числовим еквівалентом тексту в українській абетці, а саме повідомлення НІ.

Ефективність. Процедура вибору великих простих p і q описана в пункті IV.2.7. Зауважимо, що при $p \equiv q \equiv 3 \pmod 4$ алгоритм дешифрування буде особливо простим (випадок 1 алгоритму з пункту IV.4.2).

Надійність. Зрозуміло, що задача розкриття системи Рабіна, тобто знаходження за C такого M, що $E(M) = C$, є нічим іншим, як задачею добування квадратного кореня за модулем $n = pq$. В пункті IV.4.3 показано, що остання є такою ж складною, як задача факторизації числа $n = pq$ (яка, до речі, у нашому випадку є задачею знаходження таємного ключа за відкритим). Див. також вправу 3.4.

ВПРАВИ

3.1. Нехай $p = 59$ і $q = 67$.

a) Зашифрувати повідомлення ДЕСЯТЬ.

b) Розшифрувати криптотекст 0753 2556.

3.2. Нехай x — квадратичний лишок за модулем $n = pq$, де n є цілим Блюма, тобто p і q — різні прості з властивістю $p \equiv q \equiv 3 \pmod 4$. Довести, що кожен із чотирьох квадратних коренів з x в $\mathbb{Z}_n$ однозначно визначається парою бітів b_1, b_2, які для $y \in \mathbb{Z}_n^*$ рівні

$$b_1 = \begin{cases} 1, & \text{якщо } \left(\frac{y}{n}\right) = 1 \\ 0 & \text{інакше} \end{cases}, \quad b_2 = \begin{cases} 1, & \text{якщо } y \text{ — непарне ціле} \\ 0 & \text{інакше} \end{cases}.$$

3.3. Довести, що відображення $E(x) = x^2 \bmod n$, для $n = pq$ — цілого Блюма, є бієкцією з $\mathcal{Q}_n$ на $\mathcal{Q}_n$.

3.4. Показати, що система Рабіна нестійка до атаки з вибраним криптотектом.

§ 4. Ймовірнісне криптування

4.1. Ідея. Шафі Гольдвассер і Сільвіо Мікелі [99] запропонували ймовірнісну модифікацію криптографічної схеми з відкритим ключем (див. § 1). Їх ідея полягає в тому, щоб зробити алгоритм шифрування E ймовірнісним.

Ймовірнісний алгоритм шифрування ставить у відповідність повідомленню M не один криптотекст C, а деяку *родину* криптотекстів $\mathcal{C}_M$, причому кожен член $C \in \mathcal{C}_M$ цієї родини вибирається з певною ймовірністю. Іншими словами, для кожного повідомлення M результат роботи алгоритму $E(M)$ є випадковою величиною, розподіленою на множині $\mathcal{C}_M$. Щоб уможливити дешифрування, для будь-якої пари різних повідомлень M_1 та M_2 відповідні їм множини криптотекстів $\mathcal{C}_{M_1}$ та $\mathcal{C}_{M_2}$ не повинні перетинатися. Більше того, має бути ефективний алгоритм дешифрування, який використовує таємний ключ і за будь-яким $C \in \mathcal{C}_M$ визначає M.

Таку криптосистему вважають надійною, якщо для будь-якої пари повідомлень M_1 та M_2 однакової довжини l, випадкові величини $E(M_1)$ та $E(M_2)$ неможливо відрізнити одну від одної ніяким ймовірнісним поліноміальним алгоритмом із ймовірністю, вищою за $1/l$. Тобто лише

з незначною ймовірністю за двома криптотекстами можна визначити, відповідають вони одному й тому ж повідомленню чи різним. Такий рівень надійності в принципі недосяжний для детермінованих систем, у випадку яких пара різних криптотекстів C_1 і C_2 завжди відповідає різним відкритим текстам M_1 і M_2. Ймовірнісне криптування вирішує також проблему, зауважену нами на початку пункту 2.4, що детерміноване шифрування не маскує належним чином деякі з повідомлень, наприклад 0 та 1 у випадку системи RSA.

Зупинимось на понятті надійності ймовірнісної криптосистеми з відкритим ключем докладніше. Нехай A — ймовірнісний алгоритм, який завжди подає на вихід одне із двох значень — 1 або 2. Неформально це треба розуміти так, що отримавши на вхід криптотекст C, алгоритм A намагається з'ясувати, що саме має місце, $C \in \mathcal{C}_{M_1}$ чи $C \in \mathcal{C}_{M_2}$. Результат роботи A на вході C позначимо через $A(C)$.

Означення 4.1. Ймовірнісну криптосистему з відкритим ключем вважаємо *надійною*, якщо для будь-якого ймовірнісного поліноміального алгоритму A, для будь-якої пари повідомлень M_1 і M_2 однакової досить великої довжини l, ймовірності того, що $A(E(M_i)) = 1$, для $i = 1$ та $i = 2$ відрізняються не більш, ніж на $1/l$, тобто

$$|\mathbf{P}\left[A(E(M_1)) = 1\right] - \mathbf{P}\left[A(E(M_2)) = 1\right]| \leq \frac{1}{l}.$$

Ймовірність вимірюється за вибором випадкових послідовностей як алгоритмом A, так і алгоритмом шифрування E.

Таким чином, ймовірнісне криптування можна розцінювати як добре наближення до ідеалу надійності у теоретико-інформаційному сенсі, що досягається для шифру одноразового блокноту. Різниця полягає в тому, що йдеться все ж про надійність в обчислювальному сенсі — це ціна, яку доводиться платити за прийнятну, у порівнянні з шифром одноразового блокноту, довжину ключа.

4.2. Реалізація на основі квадратичності. У [99] ідея ймовірнісного криптування була втілена на основі задачі розпізнавання квадратичних лишків (див. пункти IV.1.2 і IV.4.3).

Введемо позначення $\mathbb{J}_n$ для підмножини групи $\mathbb{Z}_n^*$, що об'єднує елементи x із властивістю $\left(\frac{x}{n}\right) = 1$. Нагадаємо, що у випадку коли $n = pq$ є добутком двох різних простих, множина квадратичних лишків $\mathcal{Q}_n$ є власною підмножиною в $\mathbb{J}_n$. Множина $\widetilde{\mathcal{Q}}_n = \mathbb{J}_n \setminus \mathcal{Q}_n$ називається множиною *псевдоквадратів* за модулем n, і має рівно стільки ж елементів, що й $\mathcal{Q}_n$ (пункт **a** вправи IV.1.15). Перейдемо до опису криптосистеми.

Генерування ключів. Вибирають великі прості числа p та q, і обчислюють їх добуток $n = pq$. Вибирають випадковий псевдоквадрат $a \in \widetilde{\mathcal{Q}}_n$. Покладають

Відкритий ключ: n, a.

Таємний ключ: p, q.

Шифрування. Двійкове повідомлення $M = m_1 m_2 \dots m_l$, де $m_i \in \{0, 1\}$, перетворюють у криптотекст вигляду $C = c_1 c_2 \dots c_l$, де $c_i \in \mathbb{J}_n$. Для $i = 1, \dots, l$ елемент c_i генерують за допомогою такої ймовірнісної процедури:

- вибирають випадковий елемент $r_i \in \mathbb{Z}_n^*$ (для кожного i незалежно від всіх інших);
- для $m_i = 0$ покладають $c_i = r_i^2 \bmod n$,
- для $m_i = 1$ покладають $c_i = a r_i^2 \bmod n$.

Дешифрування. За криптотекстом $C = c_1 c_2 \dots c_l$ відкритий текст $M = m_1 m_2 \dots m_l$ визначають за таким правилом: для $i = 1, \dots, l$

$$m_i = \begin{cases} 0, & \text{якщо } c_i \in \mathcal{Q}_n, \\ 1, & \text{якщо } c_i \in \widetilde{\mathcal{Q}}_n. \end{cases}$$

Коректність. Очевидно, що бітові повідомлення $m_i = 0$ у криптотексті відповідає квадратичний лишок c_i. Бітові $m_i = 1$ відповідає елемент c_i, який є добутком квадратичного лишка і псевдоквадрату, що завжди є псевдоквадратом (див. пункт **b** вправи IV.1.15). Отже, повідомлення M однозначно визначається криптотекстом C незалежно від випадкових виборів елемента r_i алгоритмом шифрування.

Ефективність. Зупинимось на виборі псевдоквадрату a, компоненти відкритого ключа. В $\mathbb{Z}_p^*$ вибирають випадковий квадратичний нелишок a_1, а в $\mathbb{Z}_q^*$ — випадковий нелишок a_2 (див. пункт IV.4.1). Елемент a визначають з умов $a \equiv a_1 \pmod p$ і $a \equiv a_2 \pmod q$ за алгоритмом з Китайської теореми про остачі. Таке a є квадратичним нелишком, бо нелишками є і a_1, і a_2 — достатньою була б навіть одна з цих причин. З іншого боку, маємо

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p}\right)\left(\frac{a}{q}\right) = \left(\frac{a_1}{p}\right)\left(\frac{a_2}{q}\right) = (-1)(-1) = 1.$$

Отже, a є псевдоквадратом. Неважко зрозуміти, що a є випадковим елементом із $\widetilde{\mathcal{Q}}_n$, тобто розподілене на цій множині рівномірно.

При дешифруванні треба вміти відрізняти квадратичні лишки від псевдоквадратів. Для цього можна використати зведення задачі розпізнавання квадратичних лишків за модулем $n = pq$ до факторизації числа n з пункту IV.4.3, оскільки співмножники p і q даються таємним ключем.

Приклад 4.2. Візьмемо $p = 7$ і $q = 11$. Обчислимо $n = 77$. Щоб отримати a, вибираємо $a_1 = 3$ і $a_2 = 2$. Для них $a = 24$. Формування ключів завершено.

Припустимо, що ми хочемо зашифрувати повідомлення НІ.[1] Його двійковим еквівалентом є 010001 001011 (див. пункт I.3.1). Почнемо шифрування з першого біта $m_1 = 0$. Нехай $r_1 = 26$. Тоді $c_1 = 26^2 \bmod 77 = 60$. Наступним бітом є $m_2 = 1$. Нехай $r_2 = 41$. Тоді $c_2 = 24 \cdot 41^2 \bmod 77 = 73$. Продовжуємо в такому ж дусі. Можливий результат наведено в таблиці.

i	1	2	3	4	5	6	7	8	9	10	11	12
r_i	26	41	61	30	53	45	32	29	48	13	54	39
c_i	60	73	25	53	37	13	23	71	10	15	68	6

Отримуємо криптотекст 60 73 25 53 37 13 23 71 10 15 68 06.

Надійність. Зрозуміло, що задача знаходження біта m_i відкритого тексту за компонентою c_i криптотексту є задачею розпізнавання квадратичних лишків за модулем $n = pq$, для якої невідомо жодного ефективного алгоритму (без використання множників p і q, див. зауваження IV.4.5). Для $m_i = 0$ елемент c_i є випадковим квадратичним лишком, а для $m_i = 1$ випадковим псевдоквадратом за модулем n. Останнє випливає з пункту b вправи IV.1.15. На сьогодні невідомо ймовірнісного поліноміального алгоритму, який би відрізняв випадковий квадратичний лишок від випадкового псевдоквадрату за модулем n з імовірністю кращою, ніж $1/\log n$. Більше того, в [99] показано, що якби такий алгоритм існував, його можна було б перетворити у ймовірнісний алгоритм, який розпізнавав би квадратичні лишки за поліноміальний час. Таким чином, надійність криптосистеми еквівалентна важкості розпізнавання квадратичних лишків за модулем $n = pq$.

4.3. Реалізація на основі RSA функції. *Генерування ключів* відбувається таким же чином, як і для системи RSA.

[1] Це україномовне повідомлення. Не плутати з неформальним англомовним привітанням.

Відкритий ключ: e, n, де $n = pq$, НСД $(e, \phi(n)) = 1$.
Таємний ключ: d таке, що $ed \equiv 1 \pmod{\phi(n)}$.

Шифрування. Двійкове повідомлення $M = m_1 m_2 \dots m_l$, де $m_i \in \{0,1\}$, перетворюють у криптотекст виду $C = c_1 c_2 \dots c_l$, де $c_i \in \mathbb{Z}_n$. Для $i = 1, \dots, l$ елемент c_i генерують за допомогою такої ймовірнісної процедури.

- Якщо $m_i = 0$, то в $\mathbb{Z}_n$ вибирають випадкове парне число x_i; якщо $m_i = 1$, то вибирають випадкове непарне x_i.
- Обчислюють $c_i = x_i^e \bmod n$.

Дешифрування. За криптотекстом $C = c_1 c_2 \dots c_l$ відкритий текст $M = m_1 m_2 \dots m_l$ визначають за таким правилом: для $i = 1, \dots, l$

$$m_i = \begin{cases} 0, & \text{якщо } c_i^d \bmod n \text{ парне,} \\ 1, & \text{якщо } c_i^d \bmod n \text{ непарне.} \end{cases}$$

В [67] доведено, що надійність цієї криптосистеми рівносильна припущенню, що RSA не може бути розкрита поліноміальним алгоритмом.

ВПРАВИ

4.1. Довести, що якщо в означенні ймовірнісної криптосистеми з відкритим ключем обмеження на ймовірність $1/l$ поміняти на $1/l^c$ для довільної константи c, то отримаємо рівносильне поняття надійності.

4.2. а) Сформувати відкритий і таємний ключі для системи ймовірнісного криптування на основі квадратичності і зашифрувати повідомлення ТАК.

b) Використовуються ті ж ключі, що у прикладі 4.2. Криптотексти 37 64 58 23 71 67 і 25 67 60 15 53 16 76 відповідають одному й тому ж відкритому текстові чи різним?

4.3. Обгрунтувати коректність та ефективність ймовірнісного криптування на основі RSA функції.

§ 5. Система ЕльГамала

Ця ймовірнісна криптосистема була запропонована ЕльГамалом в [88].

Генерування ключів. Вибирають велике просте p, а також число g, $1 < g < p-1$, яке має в мультиплікативній групі $\mathbb{Z}_p^*$ великий порядок. В ідеальному випадку g є первісним коренем за модулем p. Числа p і g не є таємницею і перебувають в загальному користуванні. Кожен абонент вибирає собі випадкове число a у проміжку від 1 до $p-1$, і обчислює $h = g^a \bmod p$.

Відкритий ключ: p, g, h.
Таємний ключ: a.

Шифрування відбувається блоками. Кожен блок M вважаємо елементом із $\mathbb{Z}_p^*$. Повідомлення $M \in \mathbb{Z}_p^*$ перетворюють у криптотекст $C \in (\mathbb{Z}_p^*)^2$ наступним чином.

- Вибирають випадкове число r таке, що $1 \le r \le p-1$.
- Обчислюють $C = (c_1, c_2)$, де

$$c_1 = g^r \bmod p, \quad c_2 = Mh^r \bmod p.$$

Дешифрування. Маючи таємний ключ a і криптотекст $C = (c_1, c_2)$, обчислюють

$$D(C) = c_2 \cdot (c_1^a)^{-1} \bmod p.$$

Приклад 5.1. Як і в усіх попередніх прикладах, ми жертвуємо реалізмом задля простоти обчислень. Нехай $p = 23$, $g = 5$, $a = 6$. Обчислюємо $h = 5^6 \bmod 23 = 8$. Відкритий і таємний ключі сформовано.

Припустимо, що шифрується числова інформація, і потрібно зашифрувати повідомлення $M = 7$. Нехай вибрано $r = 10$. Тоді $c_1 = 5^{10} \bmod 23 = 9$ і $c_2 = (7 \cdot 8^{10}) \bmod 23 = 21$. Отримуємо криптотекст $C = (9, 21)$. Що стосується дешифрування, то легко перевірити, що справді $D(9, 21) = 21 \cdot (9^6)^{-1} \bmod 23 = 7$.

Коректність. Перевірка рівності $D(C) = M$ для криптотексту C, отриманого з повідомлення M за допомогою алгоритму шифрування з довільним r, проводиться безпосередньо і залишається читачеві в якості простої вправи. Ідея криптосистеми досить прозора: повідомлення M маскується, набираючи вигляду c_2, а разом з тим посилається підказка c_1, яка дозволяє відтворити M за c_2.

Ефективність. Піднесення до степеня в $\mathbb{Z}_p^*$ виконується за допомогою бінарного методу (пункт III.2.2). Як вибрати велике просте p, описано в пункті IV.2.7. Однак наше завдання складніше — слід вибрати також число g. Найкраще було би мати в якості g первісний корінь за модулем p. На жаль, з пункту IV.6.2 нам відомо, що знаходження первісного кореня за заданим p не є легкою задачею. Тому слід ставити завдання генерування *пари* p, g, для вирішення якого є ефективні процедури (див. пункт IV.6.2, а також вправу VII.2.6).

Надійність. Сформулюємо задачу

Розкриття системи ЕльГамала
Задано: p, g, h, c_1, c_2, де $1 < g < p-1$,
$h = g^a \bmod p$, $c_1 = g^r \bmod p$, $c_2 = Mh^r \bmod p$
для деяких $a, r, M \in \mathbb{Z}_p^*$.
Обчислити: M.

Черговою простою вправою для читача є встановлення того, що ця задача еквівалентна наступній.

Розкриття системи ЕльГамала — 2
Задано: $p, g, x, y \in \mathbb{N}$, де $1 < g < p-1$, $x = g^a \bmod p$,
$y = g^b \bmod p$ для деяких a і b.
Обчислити: $z = g^{ab} \bmod p$.

Для другої задачі очевидно, що вона зводиться до дискретного логарифмування за модулем p. Тому слід виключити ті випадки, у яких логарифмування можна провести ефективно. Зокрема, p слід вибирати так, щоб число $p-1$ мало великий простий дільник, інакше суперник зможе скористатися алгоритмом Сільвера-Поліга-Гелмана (пункт IV.7.2). В [80] показано, що у випадку, коли $\phi(p-1)$ має лише малі прості дільники, задача розкриття криптосистеми ЕльГамала еквівалентна дискретному логарифмуванню.

ВПРАВИ

5.1. а) Нехай $p = 17$, $g = 3$. Вибрати a і завершити формування ключів. Зашифрувати числове повідомлення 5, вибравши r на власний розсуд.

b) Нехай $p = 17$, $g = 3$, $a = 7$. Розшифрувати криптотекст $C = (5, 15)$.

5.2. Нехай $M \in \mathbb{Z}_p^*$ — довільне повідомлення, що шифрується в системі ЕльГамала. Припустимо, що g є первісним коренем за модулем p.

a) Чи c_1 розподілене рівномірно на $\mathbb{Z}_p^*$?

b) Нехай $s \mid (p-1)$ і $a = (p-1)/s$. Скільки значень може набувати c_2? (При малих s приклад поганого вибору a.)

c) Для яких фіксованих a компонента криптотексту c_2 розподілена рівномірно на $\mathbb{Z}_p^*$? Скільки є таких a? (Приклад доброго вибору a.)

d) Чи c_2 розподілене рівномірно на $\mathbb{Z}_p^*$ у припущенні, що таємний ключ a не фіксований, а є випадковим елементом множини $\mathbb{Z}_p^*$?

5.3. Нехай g є первісним коренем за модулем p.

a) Придумати процедуру, яка ламає систему ЕльГамала, а саме знаходить повідомлення за криптотекстом, у частковому випадку, коли $r = (p-1)/2$.

b) Те ж у випадку, коли $3 \mid (p-1)$ і $r = (p-1)/3$ або $r = 2(p-1)/3$.

c) Позначимо $s = (p-1)/\text{НСД}(r, p-1)$. Придумати процедуру, яка ламає систему ЕльГамала за час, обмежений поліномом від довжини повідомлення і величин $\log p$ та s.

Розділ VI.

Криптографічні інструменти

§ 1. Важкооборотні функції

1.1. Означення і приклади. Як і досі, ми розглядаємо функції, областю визначення і множиною значень яких є множина слів у деякому алфавіті. Без втрати загальності обмежимось розглядом двійкового алфавіту. Функція f називається *важкооборотною*, якщо виконуються такі дві умови.

1) f ефективно обчислюється.
2) Жоден ефективний алгоритм для більшості аргументів $x \in \{0,1\}^n$ невзмозі за образом $y = f(x)$ знайти ніякого елемента x' такого, що $f(x') = y$.

Для практика таке означення є цілком зрозумілим. Прикладом може бути функція $DES : \{0,1\}^{64} \to \{0,1\}^{64}$, якій було присвячено пункт I.3.4. Ця функція є бієктивною і умова 2 означає просто, що обернена функція не може бути обчисленою ефективно для більшості входів — твердження, яке власне й означає надійність системи DES.

Математична теорія пропонує формалізацію цього, з погляду практика, самодостатнього поняття. Умова 1 легкості обчислення функції f і умова 2 важкості її обернення стають абсолютно строгими. Платою за математичну строгість є те, що ці умови набувають асимптотичного характеру, внаслідок чого стають непридатними для характеризації конкретних скінченних функцій. Втім, така ситуація є типовою для теорії складності, яка попри все добре узгоджується із програмістською практикою.

Одне з можливих означень є таким.

Означення 1.1. Функція f називається *важкооборотною*, якщо

1) f обчислюється за поліноміальний час.
2) Кожен поліноміальний ймовірнісний алгоритм на вході $y = f(x)$ для випадкового $x \in \{0,1\}^n$ знаходить якийсь із проообразів значення y із ймовірністю, що для досить великих n не перевищує $1/n$. Ймовірність тут поширюється на випадковий вибір слова $x \in \{0,1\}^n$ і на випадкову послідовність ймовірнісного алгоритму.

З огляду на вступний характер нашого викладу, ми не зупиняємось на нюансах та різновидах означення важкооборотної функції. Зацікавлений читач скеровується до [93].

Ні для якої конкретної функції не доведено, що вона важкооборотна. Але є функції, які гіпотетично вважаються такими і з успіхом застосовуються на практиці.

Функція множення: $MULT(x,y) = xy$.
Функція визначена на парах натуральних чисел x і y з однаковою кількістю значущих двійкових цифр.

RSA функція: $RSA(x,e,m) = \langle x^e \bmod m, e, m\rangle$.
Визначена для $m = pq$, що є добутком двох різних простих, для e такого, що НСД $(e, \phi(m)) = 1$, і для $x \in \mathbb{Z}_m$. Для фіксованих m і e є бієкцією $\mathbb{Z}_m$ на себе, або інакше кажучи, перестановкою множини $\mathbb{Z}_m$ (існування оберненого відображення випливає із твердження V.2.2).

Функція Рабіна (квадратична функція):
$SQUARE(x,m) = \langle x^2 \bmod m, m\rangle$.
Визначена для $m = pq$ і $x \in \mathbb{Z}_m$, де m є цілим Блюма, тобто p і q — різні прості з властивістю $p \equiv q \equiv 3 \pmod 4$. Для фіксованого m звуження на $\mathcal{Q}_m$ є перестановкою цієї множини (див. вправу V.3.3).

Експоненційна функція: $EXP(x,g,p) = \langle g^x \bmod p, g, p\rangle$.
Визначена для довільного простого p, первісного кореня g за модулем p, і $x \in \mathbb{Z}_p^*$. Для фіксованих p і g є перестановкою множини $\mathbb{Z}_p^*$ (як легко випливає із вправи IV.1.4).

Функції RSA і Рабіна є кандидатами у *важкооборотні функції із секретом*. Крім вищеперелічених умов 1 і 2 ці функції задовольняють ще й третю: володіння деяким секретом (а саме, співмножниками p і q) дозволяє ефективно обчислювати обернені функції.

1.2. Перші застосування. Важкооборотні функції є потужним криптографічним інструментом. На їх основі можна конструювати різні

системи ймовірнісного криптування. При це йтиметься далі, а зараз наведемо просте, але ефектне застосування.

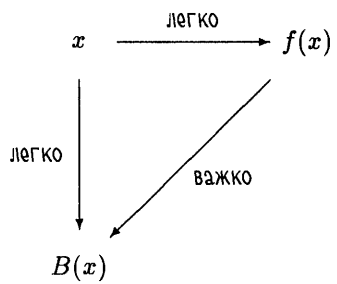
Щоб отримати доступ до комп'ютерної мережі, законний користувач повинен засвічити свою особу, ввівши відомий лише йому *пароль* або *гасло* (обидва терміни є синонімами, ми надаємо перевагу другому). Несанкціонований користувач, що добре знає систему, при нагоді може знайти в пам'яті місце зі списком гасел легальних користувачів, і надалі користуватися якимось із них. Щоб виключити таку можливість, системі достатньо зберігати у списку не гасло x, а його образ $y = f(x)$ відносно деякої важкооборотної функції f. При введенні гасла x перевірка рівності $f(x) = y$ здійснюватиметься швидко. З іншого боку, зловмисник, який натрапить на y, не зможе знайти ніякого x' із властивістю $f(x') = y$ завдяки важкооборотності функції f. Як завжди у наших заняттях криптологією, ми цілком припускаємо, що зловмисник знає алгоритм обчислення функції f — це нічим не полегшить його завдання.

Ще одне застосування важкооборотної функції корисне в наступній ситуації. Уявимо собі криптосистему з відкритим ключем для багатьох користувачів на кшталт системи ЕльГамала. Припустимо, що спільною компонентою відкритого ключа є просте число p, яке вибирається і оприлюднюється менеджером центру розподілу ключів. Нечесний менеджер може вибрати p не випадковим чином, а якось так, що отримає можливість читати пошту абонентів своєї мережі (наприклад, він знатиме розклад числа $p-1$ на прості множники, причому вони будуть малими — цього досить для зламу системи ЕльГамала).

В плані такої загрози абоненти зможуть почувати себе впевненіше, якщо менеджер разом із p пред'явить їм x, для якого $f(x) = p$, де f — деяка важкооборотна перестановка множини чисел однакової довжини. Виконання цієї додаткової вимоги не є надто обтяжливим. При генеруванні простого числа менеджер вибирає випадкове x і замість тестування його на простоту, тестує $f(x)$. Зауважимо, що $f(x)$ — випадкове число однакової з x довжини, тому ймовірність успішного знаходження простого числа потрібної довжини залишається такою ж. З іншого боку, якщо менеджер добув p якимось підступним способом, то йому буде важко знайти x з властивістю $f(x) = p$ з огляду на важкооборотність функції f.

Ми виходили з припущення, що f є бієкцією. Однак з подібною метою можна використати довільну важкооборотну функцію. Конкретна реалізація цієї ідеї входить в стандарт DSS [122].

1.3. Поняття ядра функції. Повертаючись до означень важкооборотних функцій *RSA*, *SQUARE* та *EXP*, бачимо, що після фіксації відповідних параметрів, ці функції стають перестановками елементів деякої множини D. Маючи на увазі ці приклади, розглянемо ефективно обчислюване бієктивне відображення $f : D \to D$. Водночас розглянемо ефективно обчислюваний предикат $B : D \to \{0,1\}$ — для кожного $x \in D$ можна легко обчислити біт $B(x)$. Нехай $f(x) = y$. Оскільки f — бієкція, то $B(x)$ однозначно визначається елементом y. Однак це не означає, що маючи y, отримати $B(x)$ легко. Якщо немає ефективного алгоритму, який би для більшості елементів $x \in D$ за заданим значенням $f(x)$ давав біт $B(x)$, то предикат B називається *ядром* функції f (див. малюнок 1).



Малюнок 1. Важкооборотна функція f і її ядро B.

Нескладно збагнути, що коли функція f має ядро, то вона важкооборотна. Справді, якби був ефективний алгоритм для обчислення оберненої функції f^{-1}, то $B(x)$ було би легко отримати з y композицією алгоритмів для обчислення f^{-1} і B.

Тепер дамо формальне означення. Нехай D_n — послідовність множин, для якої $n = \lceil \log \|D_n\| \rceil$. Нижче поліноміальна обчислюваність означає можливість обчислення за час, обмежений поліномом від n.

Означення 1.2. Поліноміально обчислюваний предикат $B : D_n \to \{0,1\}$ називається *ядром* функції $f : D_n \to D_n$, якщо будь-який поліноміальний ймовірнісний алгоритм на вході $f(x)$, де x — випадковий елемент із D_n, видає значення $B(x)$ з імовірністю меншою, ніж $1/2+1/n^c$, для довільної константи c при досить великих n. Ймовірність береться як за випадковим вибором x, так і за випадковою послідовністю ймовірнісного алгоритму.

По суті, означення говорить, що немає ніякого кращого способу отримати $B(x)$ за $f(x)$, ніж просто взяти навмання випадковий біт.

Для перелічених у попередньому пункті важкооборотних функцій в якості ядра запропоновано такі предикати.

Для функції $RSA : \mathbb{Z}_m \to \mathbb{Z}_m$ з фіксованими параметрами e і m, а також для функції $SQUARE : \mathcal{Q}_m \to \mathcal{Q}_m$ з фіксованим параметром m, B є предикатом парності, тобто $B(x) = 1$ для непарних x і лише для них. Для функції $EXP : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ з фіксованими параметрами p і g предикат B задається співвідношенням

$$B(x) = \begin{cases} 1, & \text{якщо } x \le (p-1)/2; \\ 0, & \text{інакше.} \end{cases}$$

Застосування важкооборотних функцій та їх ядер для генерування послідовностей псевдовипадкових бітів буде описане у пункті 2.3. А в наступному пункті висвітлюється зв'язок цих понять з імовірнісним криптуванням.

1.4. Предикат із секретом і ймовірнісне криптування. Повернемося до малюнка 1, де f — перестановка множини D_n, а B — її ядро. Вважаємо, що множина D_n має природну структуру, а саме, існує поліноміальний ймовірнісний алгоритм, який на вході $1^n = 11\ldots1$ (параметр n в унарному записі) видає випадковий елемент x, рівномірно розподілений на D_n. Припустимо, що f — важкооборотна функція з секретом. Прикладом може служити функція RSA на множині $\mathbb{Z}_m$ (при фіксації параметрів m і e). Розглянемо предикат $B' : D_n \to \{0,1\}$, заданий співвідношенням $B'(y) = B(f^{-1}(y))$. З означень важкооборотної функції з секретом і її ядра випливають такі властивості.

1) Будь-який ймовірнісний поліноміальний алгоритм на випадковому вході $y \in D_n$ видає правильне значення $B'(y)$ з імовірністю не кращою, ніж $1/2 + 1/n^c$.
2) Знання секрету дозволяє легко обчислити $B'(y)$ для довільного $y \in D_n$.
3) Є поліноміальний алгоритм, який, отримавши на вхід слово 1^n і біт $b \in \{0,1\}$, видає елемент y, рівномірно розподілений на множині $\{y \in D_n : B'(y) = b\}$.

Остання властивість виконується завдяки тому, що f — бієкція. Алгоритм вибирає випадковий елемент $x \in D_n$ і якщо $B(x) = b$, то подає на вихід $y = f(x)$, а інакше пробує інший випадковий x.

Означення 1.3. Предикат B' із властивостями 1–3 називають *предикатом із секретом*.

Маючи якийсь предикат із секретом, можна влаштувати ймовірнісне криптування, надійне в сенсі означення V.4.1. Щоб зашифрувати двійкове повідомлення $M = m_1 \ldots m_l$, для кожного $i \le l$ слід вибрати випадковий елемент $y_i \in D_n$ такий, що $B'(y_i) = m_i$. Криптотекстом буде послідовність $y_1 \ldots y_l$. Надійність такої системи ймовірнісного криптування забезпечується завдяки умові 1 вище, причому n виступає параметром надійності. Відкритим ключем цієї криптосистеми є дані, потрібні для специфікації алгоритму з умови 3. Таємний ключ складається із секретних параметрів для обчислення предикату B', яке здійснюється при дешифруванні. Існування цих параметрів забезпечене умовою 2.

Схема ймовірнісного криптування на базі функції RSA із пункту V.4.3 є конкретною реалізацією описаної конструкції.

ВПРАВИ

1.1. Пояснити, чому предикат парності не є ядром для функції EXP.

1.2. Для розпізнавання квадратичних лишків за модулем m, де m — ціле Блюма, невідомо поліноміального ймовірнісного алгоритму. Виходячи з припущення, що такого алгоритму не існує, довести, що предикат парності є ядром функції Рабіна.

1.3. Позначимо через $f_{e,m} : \mathbb{Z}_m \to \mathbb{Z}_m$ перестановку, яка отримується з функції RSA фіксацією параметрів e та m. Нехай $P(x) = x \bmod 2$ — предикат парності, а предикат Q задається на $\mathbb{Z}_m$ співвідношенням

$$Q(x) = \begin{cases} 0, & \text{якщо } 0 \le x < m/2; \\ 1, & \text{якщо } m/2 < x \le m-1. \end{cases}$$

Розглянемо відповідні два предикати із секретом:

$$B_1(e,m,y) = P(f_{e,m}^{-1}(y)) \quad \text{та} \quad B_2(e,m,y) = Q(f_{e,m}^{-1}(y)).$$

Довести, що задачі обчислення цих предикатів є поліноміально еквівалентними.

1.4. Показати, що розкриття криптосистеми RSA поліноміально зводиться до обчислення предикату B_1 з попередньої задачі. Таким чином, якщо є ефективний спосіб за криптотекстом і відкритим ключем визначати лише один, а саме наймолодший, біт відповідного відкритого тексту, то систему RSA можна зламати. Те ж саме можна сформулювати й так: якщо система RSA надійна, то неможливо визначити навіть один (наймолодший) біт повідомлення.

1.5. Довести надійність схеми ймовірнісного криптування на основі предикату з секретом.

ЛІТЕРАТУРА

Про важкооборотні функції, що грунтуються на важкості задач алгебраїчного кодування, можна дізнатися з [52, 93, 136].

Аналіз важкооборотних функцій на предмет виділення їх ядра був проведений у [148, 67] для *RSA*, у [77, 67] для *SQUARE*, і у [79] для *EXP*. В цих роботах доведено, що коли перелічені функції є важкооборотними, то відповідні предикати є їх ядрами. У [150] встановлено, що кожна важкооборотна функція має ядро. Значно простішу конструкцію ядра запропоновано в [96].

Ідея використання предикатів із секретом для ймовірнісного криптування з'явилась у [99].

§ 2. Генератори псевдовипадкових бітів

2.1. Означення генератора псевдовипадкових послідовностей. Упродовж попередніх розділів ми використовували випадкові послідовності бітів із двоякою метою: по-перше, у ймовірнісних алгоритмах, і по-друге, у криптосистемах. Звернемо увагу, що випадкові послідовності застосовувалися не лише у ймовірнісних криптосистемах, де це робилося цілком явно, але й у класичних криптосистемах, де таємний ключ слід було вибирати випадковим чином. Типовим прикладом є шифр одноразового блокноту.

Нагадаємо, що випадковою послідовністю бітів довжини n називається випадкова величина, яка набуває кожне значення із $\{0,1\}^n$ з однаковою ймовірністю 2^{-n}. Для невеликого n випадкову послідовність отримати нескладно, наприклад, підкиданням монетки. Однак породження довгої випадкової послідовності бітів є вельми нетривіальним завданням для обчислювальної машини.

Цей розділ нашого курсу присвячений такому питанню. Припустимо, що ми вміємо генерувати випадкову двійкову послідовність довжини n. Яким чином її можна розширити до випадкової у певному сенсі двійкової послідовності довжини n^d, де d є деякою константою? Під словом "розширити" маємо на увазі "перетворити за допомогою детермінованого алгоритму". Зрозуміло, що внаслідок такого розширення ми не зможемо отримати справжню випадкову послідовність довжини n^d хоча б тому, що серед всіх 2^{n^d} таких послідовностей з ненульовою

ймовірністю з'являтимуться щонайбільше 2^n. Однак хотілося б отримати принаймні *псевдо*випадкову послідовність довжини n^d, тобто послідовність, яка виглядала б як випадкова. Метою серії означень нижче є уточнити і формалізувати останню фразу.

Означення 2.1. *Ансамблем* будемо називати послідовність випадкових величин Z_n, де $n \in \mathbb{N}$, які набувають значення у множині двійкових слів.

Означення 2.2. Ансамблі $\{X_n\}_{n\in\mathbb{N}}$ і $\{Y_n\}_{n\in\mathbb{N}}$ називаються *нерозрізнюваними* за поліноміальний час, якщо для будь-якого поліноміального ймовірнісного алгоритму A

$$|\mathbf{P}[A(X_n)=1]-\mathbf{P}[A(Y_n)=1]|<\frac{1}{n^c},$$

для довільної константи c при досить великих n. Тут $\mathbf{P}[A(Z_n)=1]$ позначає ймовірність того, що алгоритм A, отримавши на вхід випадкову величину Z_n, подасть на вихід 1. Ймовірність береться як за розподілом Z_n, так і за розподілом випадкової послідовності ймовірнісного алгоритму A.

Означення 2.3. Нехай $l : \mathbb{N} \to \mathbb{N}$ — функція така, що $l(n) > n$. *Генератором із розширенням* l називається поліноміальний детермінований алгоритм G, який двійкову послідовність $x \in \{0,1\}^n$ перетворює у послідовність $G(x) \in \{0,1\}^{l(n)}$.

Випадкову послідовність x, що подається на вхід генератора, називатимемо *паростком*. Задля технічної зручності домовимось, що для деяких паростків x генератор G може видавати відмову, і тоді $G(x)$ буде невизначеним.

Тепер домовимось, які генератори заслуговують бути названими *генераторами псевдовипадкових послідовностей* або, коротше, *псевдовипадковими генераторами*. Визначальною рисою такого генератора $G : \{0,1\}^n \to \{0,1\}^{l(n)}$ має бути те, що випадкову величину $G(x)$, де x — випадковий паросток із $\{0,1\}^n$, ніяким поліноміальним ймовірнісним алгоритмом не можна відрізнити від випадкової величини, рівномірно розподіленої на $\{0,1\}^{l(n)}$.

Означення 2.4. Нехай G — генератор із розширенням l. Позначимо через $Z_{l(n)}$ випадкову величину $G(x)$ зі значеннями в $\{0,1\}^{l(n)}$, для x рівномірно розподіленого на $\{0,1\}^n$. Через U_n позначимо випадкову величину з рівномірним розподілом на $\{0,1\}^n$. Генератор G

називається *псевдовипадковим*, якщо ансамблі $\{Z_{l(n)}\}_{n\in\mathbb{N}}$ і $\{U_{l(n)}\}_{n\in\mathbb{N}}$ нерозрізнювані за поліноміальний час.

Двійкову послідовність $G(x)$, отриману за допомогою псевдовипадкового генератора G із випадкового паростка x, будемо називати *псевдовипадковою*.

Здавалося б, що розширення l є важливою кількісною характеристикою генератора псевдовипадкових послідовностей. Однак, наступне твердження дещо несподівано виявляє, що величина l не відіграє надто принципової ролі — генератор з мінімальним розширенням можна перетворити в генератор з довільним поліноміальним розширенням.

Твердження 2.5. [О. Гольдрайх, С. Мікелі] *З довільного псевдовипадкового генератора G з розширенням $l(n) = n+1$ можна отримати псевдовипадковий генератор G' із розширенням $l'(n) = n^d$ для довільної цілої константи $d > 1$.*

Доведення. Винесене у вправу 2.1. ∎

Псевдовипадкові генератори цілком придатні для використання у ймовірнісних алгоритмах. Якщо алгоритм потребує випадкової послідовності довжини n^d, то замість неї можна використовувати псевдовипадкову послідовність такої ж довжини, отриману із справді випадкового паростка довжини n (див. вправу 2.3).

2.2. Непередбачуваність псевдовипадкових генераторів. У криптографічних застосуваннях від псевдовипадкового генератора G вимагається виконання такої умови. Нехай суперник спостерігає випадкову величину $G(x)$, розподілену на $\{0,1\}^{l(n)}$ і породжену рівномірним розподілом паростка x на $\{0,1\}^n$. Припустимо, що спочатку йому відкриваються лише перші i бітів величини $G(x)$. Тоді бажано, щоб суперник не міг передбачити $(i+1)$-ий біт з імовірністю, суттєво кращою за $1/2$. Іншими словами, ми зацікавлені, щоб не було кращого шляху отримати $(i+1)$-ий біт на основі знання всіх попередніх бітів, аніж простим підкиданням монетки.

Зв'язок цієї вимоги із криптографічною надійністю можна пояснити таким прикладом. Припустимо, що суперник перехопив криптотекст у двійковому алфавіті, і йому відомі такі факти:

1) вжито шифр одноразового блокноту;
2) ключ є псевдовипадковою послідовністю;
3) відкритий текст починається зверненням "Вельмишановний Іване Івановичу".

Тоді додавши відомий йому префікс відкритого тексту до відповідного префіксу криптотексту, суперник отримає перші i бітів псевдовипадкового ключа. Якщо генератор, який було використано для вибору ключа, не володіє описаною вище властивістю, то суперник зможе екстраполювати наступні біти ключа і добути із криптотексту ще більше інформації.

Означення 2.6. Нехай для кожного натурального n випадкова величина Z_n набуває значення в $\{0,1\}^n$. Ансамбль $\{Z_n\}_{n\in\mathbb{N}}$ називається *непередбачуваним* або, як ще кажуть, *витримує тестування наступного біту*, якщо виконується така умова. Припустимо, що $s\sigma$ — префікс випадкового слова Z_n, де $\sigma \in \{0,1\}$ і $s \in \{0,1\}^{k-1}$, причому довжина k цього префіксу є випадковим числом від 1 до n. Нехай A — довільний поліноміальний ймовірнісний алгоритм, який отримує на вхід s. Тоді

$$\left|\mathbf{P}\left[A(s) = \sigma\right] - \frac{1}{2}\right| < \frac{1}{n^c},$$

для довільної константи c при досить великих n.

Виявляється, що псевдовипадкові генератори і лише вони володіють властивістю непередбачуваності.

Теорема 2.7. *Нехай $G : \{0,1\}^n \to \{0,1\}^{l(n)}$ — генератор із розширенням l. Позначимо через $Z_{l(n)}$ випадкову величину $G(x)$ зі значеннями в $\{0,1\}^{l(n)}$, для x рівномірно розподіленого на $\{0,1\}^n$. Генератор G є псевдовипадковим тоді і лише тоді, коли ансамбль $\{Z_n\}_{n\in\mathbb{N}}$ непередбачуваний.* ∎

На підставі цієї теореми і в світлі попереднього обговорення псевдовипадкові генератори часом називають *криптографічно надійними*.

2.3. Псевдовипадкові генератори із важкооборотних перестановок. Нехай f — важкооборотна функція, яка після фіксації відповідних параметрів є перестановкою деякої множини D_n, причому $\lceil \log \|D_n\| \rceil = n$. Прикладами є функції *RSA*, *SQUARE* і *EXP*, які визначають перестановки множин $\mathbb{Z}_m$, $\mathcal{Q}_m$ і $\mathbb{Z}_p^*$ (в позначеннях пункту 1.1). Нехай B — ядро функції f (відповідні предикати для перелічених функцій див. у пункті 1.3). Використовуючи f і B, сконструюємо генератор G з розширенням $l(n)$, де функція $l(n)$ обмежена деяким поліномом від n. Вважаємо, що $D_n \subseteq \{0,1\}^n$, і що належність до цієї множини можна ефективно розпізнати.

Вхід: паросток $x_0 \in \{0,1\}^n$.

- Якщо $x_0 \notin D_n$, то зупинитись, інакше продовжувати.
- Для $i = 1, \ldots, l(n)$ обчислити $\sigma_i = B(x_{i-1})$ і $x_i = f(x_{i-1})$.
- Подати на вихід послідовність $\sigma_1\sigma_2\ldots\sigma_{l(n)}$.

Теорема 2.8. *Для кожної важкооборотної перестановки f і її ядра B, описаний генератор G є псевдовипадковим.* ∎

Зауваження 2.9. Теорема говорить, що якщо паросток x_0 вибирається рівноймовірно з $\{0,1\}^n$, то ніякий поліноміальний ймовірнісний алгоритм неспроможний відрізнити послідовність $\sigma_1\sigma_2\ldots\sigma_l$ від справжньої випадкової послідовності такої ж довжини. Насправді це твердження залишається в силі, навіть коли оприлюднюється останній елемент x_l, тобто, послідовність $\sigma_1\sigma_2\ldots\sigma_l x_l$ не можна відрізнити за поліноміальний час від послідовності $\rho_1\rho_2\ldots\rho_l x_l$, де $\rho_1, \ldots, \rho_l$ — випадкові і незалежні між собою біти. Зауважимо також, що x_l є випадковим елементом множини D_n.

2.4. BBS генератор. Окремо зупинимось на реалізації конструкції із попереднього пункту на основі функції Рабіна, яку ми позначаємо через *SQUARE*. Генератор, що виходить в результаті, називають генератором BBS, де абревіатура утворена від імен його винахідників — Ленори Блюм, Мануїла Блюма та Майка Шуба [77].

Нехай параметр $l = l(n)$ обмежений поліномом від n. Щоб отримати із n випадкових бітів l псевдовипадкових, слід

- вибрати випадково прості p та q такі, що $2^{n-1} < m < 2^n$ для $m = pq$, і $p \equiv q \equiv 3 \pmod 4$;
- вибрати випадковий елемент $r \in \mathbb{Z}_m^*$ і обчислити $x_0 = r^2 \bmod m$;
- для i від 1 до l обчислювати

$$\sigma_i = x_{i-1} \bmod 2, \quad x_i = x_{i-1}^2 \bmod m;$$

- подати на вихід послідовність $\sigma_1\sigma_2\ldots\sigma_l$.

Нескладно помітити, що ми притримувались загальної конструкції попереднього пункту із функцією $f(x) = x^2 \bmod m$ і її ядром, предикатом парності $B(x) = x \bmod 2$. Нагадаємо, що функція f не є бієктивною на $\mathbb{Z}_m^*$, але є такою на $\mathcal{Q}_m$. Саме тому в якості паростка ми вибирали $x_0 = r^2 \bmod m$ — випадковий елемент множини $\mathcal{Q}_m$. За теоремою 2.8 BBS генератор є псевдовипадковим, а отже, криптографічно надійним (за умови, що функція *SQUARE* справді є важкооборотною).

Приклад 2.10. Нехай $p = 19$ і $q = 23$. Тоді $m = 437$. Для $r = 233$ отримуємо

i	0	1	2	3	4	5	6	7	8	9	10	11	12
x_i	101	150	213	358	123	271	25	188	384	187	9	81	6
σ_{i+1}	1	0	1	0	1	1	1	0	0	1	1	1	0

Деякі властивості генератора роблять його використання особливо зручним.

Твердження 2.11. *За x_l і простими p та q можна ефективно визначити всю послідовність $x_0, \ldots, x_{l-1}$, а отже і послідовність $\sigma_1\ldots\sigma_l$.*

Зауважимо, що твердження не суперечить криптографічній надійності BBS генератора, оскільки в застосуваннях прості p і q тримаються у таємниці.

Доведення. Припустимо, що відомо елемент x_i, де $1 \le i \le l$, і покажемо, як отримати x_{i-1}. За бієктивністю функції $f(x) = x^2 \bmod m$ на $\mathcal{Q}_m$, елемент x_{i-1} однозначно визначається такими умовами

1) $x_{i-1}^2 \equiv x_i \pmod m$,
2) $x_{i-1} \in \mathcal{Q}_m$.

Введемо позначення $y = x_{i-1} \bmod p$ і $z = x_{i-1} \bmod q$. За наслідком II.2.13 Китайської теореми про остачі, лишки y і z задовольняють умови

$$y^2 \equiv x_i \pmod p, \qquad z^2 \equiv x_i \pmod q, \tag{1}$$

$$y \in \mathcal{Q}_p, \qquad z \in \mathcal{Q}_q, \tag{2}$$

і, більше того, така пара y, z однозначно визначає x_{i-1}, причому ефективним чином.

Отже, досить знайти y та z. Для цього використаємо співвідношення

$$y = x_i^{\frac{p+1}{4}} \bmod p \quad \text{та} \quad z = x_i^{\frac{p+1}{4}} \bmod q. \tag{3}$$

З них безпосередньо видно, що y і z ефективно обчислюються із застосуванням бінарного методу піднесення до степеня.

Нам залишилося обгрунтувати рівності (3). Доведемо першу з них (для другої аргументи ідентичні). Досить перевірити умови (1) та (2) для y, заданого співвідношенням (3). Умова (2) негайно випливає із замкнутості $\mathcal{Q}_p$ відносно множення. Умова (1) випливає з того, що добування кореня за модулем p таким, що $p \equiv 3 \pmod 4$, еквівалентне піднесенню до степеня $(p+1)/4$ — див. випадок 1 алгоритму з пункту IV.4.2.

Доведення завершено. ∎

2.5. Потокові шифри. *Потоковим* називається шифр, який двійкове повідомлення $M = m_1 m_2 \dots m_l$ з використанням двійкового ключа такої ж довжини $K = k_1 k_2 \dots k_l$ перетворює у криптотекст $C = c_1 c_2 \dots c_l$ шляхом покомпонентного додавання M і K за модулем 2: $c_i = (m_i + k_i) \bmod 2$. Останнє прийнято записувати як $c_1 = m_i \oplus k_i$, а також $C = M \oplus K$.

Різновиди потокових шифрів відрізняються між собою способом продукування ключа. У шифрі одноразового блокноту кожен біт ключа вибирається випадково і незалежно від інших бітів. Як зазначалося у пункті I.3.2, за рахунок цього досягається абсолютна надійність шифру у теоретико-інформаційному сенсі, але ця ж обставина робить шифр одноразового блокноту непрактичним для довгих повідомлень.

Виходом може бути породження ключа K за допомогою криптографічно надійного генератора псевдовипадкових бітів (при цьому ключем варто називати не всю послідовність K, а лише паросток, що використовується при її генеруванні). Дамо опис конкретної реалізації цієї ідеї, яка використовує BBS генератор.

Криптосистема Блюма-Гольдвассер [78]

Таємний ключ: p і q — великі прості числа, такі, що $p \equiv q \equiv 3 \pmod 4$.

Відкритий ключ: m, де $m = pq$.

Шифрування. Щоб зашифрувати двійкове повідомлення M довжини l, Боб формує двійкову послідовність $K = k_1 k_2 \dots k_l$ з l псевдовипадкових бітів, отриманих за допомогою BBS генератора. Для цього він попередньо вибирає випадковий квадратичний лишок x_0 за модулем m, послідовним піднесенням до квадрату отримує послідовність $x_0, \dots, x_l \in Q_m$, і покладає $k_i = x_{i-1} \bmod 2$. Далі Боб обчислює $C = M \oplus K$ і посилає Алісі криптотекст, який складається з пари (C, x_l). Зазначимо, що описане криптування є ймовірнісним, оскільки при різних виборах паростка для генерування псевдовипадкової послідовності K, повідомленню будуть відповідати різні криптотексти.

Приклад 2.12. Нехай $p = 19$ і $q = 23$ — таємний ключ. Тоді відкритим ключем буде $m = 437$. Нехай потрібно закриптувати повідомлення HI.[1] Записуємо повідомлення у двійковій формі: $M = 010001\ 001011$. Запускаємо BBS генератор, вибравши паросток $r = 233$. Ми потребуємо 12 псевдовипадкових бітів. Відповідні обчислення були проведені

[1] Див. примітку на стор. 142.

у прикладі 2.10. Маємо $K = 101011\ 100111$ і $x_{12} = 6$. Обчислюємо $C = 111010\ 101100$. Отже, криптотекстом буде пара $(111010\ 101100, 6)$.

Дешифрування. Аліса, яка знає таємний ключ p і q, за числом x_l обчислює всі члени послідовності $x_0, \dots, x_{l-1}$ (обчислення відбувається у зворотному порядку, як описано у доведенні твердження 2.11). На основі цієї послідовності Аліса відтворює двійковий вектор K і отримує $M = C \oplus K$.

Ефективність. За швидкістю реалізацій криптосистема порівнянна із системою RSA. Вона також вигідно відрізняється від системи ймовірнісного криптування з пункту V.4.2 в такому плані. В останній криптосистемі розмір криптотексту більший від розміру повідомлення у кількість разів, що дорівнює довжині ключа, в той час як у системі Блюма-Гольдвассер криптотекст довший від повідомлення лише *на* довжину ключа.

Надійність. Доведено, що система є надійною за умови важкості факторизації цілих Блюма. Один із варіантів відповідної теореми звучить так. Для будь-якого повідомлення M довжини $l = (\log m)^{O(1)}$, випадкову величину (C, x_l) неможливо відрізнити від величини $(\rho_1 \dots \rho_l, x)$, що складається із справжньої випадкової двійкової послідовності $\rho_1 \dots \rho_l$ і випадкового квадратичного лишка $x \in Q_m$, якою-небудь схемою поліноміального від l розміру (див. вправу 2.2). Доведення є нескладною комбінацією неоднорідної версії теореми 2.8 (див. також зауваження 2.9) та результатів про ядро функції Рабіна в [77, 67] (див. вправу 1.2). Зауважимо, що криптосистема із щойно сформулюваною умовою надійності є надійною в сенсі означення V.4.1.

ВПРАВИ.

2.1. Нехай $c > 1$ — ціла константа. Припустимо, що G — псевдовипадковий генератор з розширенням $n + 1$. Довести, що тоді псевдовипадковим є також генератор G' з розширенням n^d, який задається співвідношенням $G'(x) = \sigma_1 \dots \sigma_{n^d}$ для $x \in \{0,1\}^n$, де $\sigma_i x_i = G(x_{i-1})$ і $x_0 = x$.

2.2. Модифікуємо означення 2.2 нерозрізнюваності ансамблів, замінивши у ньому алгоритм A на послідовність функціональних схем C_n (див. пункт III.2.3). При цьому розглядаємо такі послідовності, що схема C_n обчислює деяку булеву функцію $f : \{0,1\}^n \to \{0,1\}$, і розмір схеми C_n обмежений деяким поліномом від n. Ансамблі $\{X_n\}_{n \in \mathbb{N}}$ і $\{Y_n\}_{n \in \mathbb{N}}$ будемо називати *поліноміально нерозрізнюваними*, якщо для будь-якої такої послідовності схем $\{C_n\}_{n \in \mathbb{N}}$ поліноміального розміру

$$|\mathbf{P}[C(X_n) = 1] - \mathbf{P}[C(Y_n) = 1]| < \frac{1}{n^c}$$

для довільної константи c при досить великих n. Нове означення будемо називати *неоднорідним*, а "старе" означення 2.2 — *однорідним* (про поняття *неоднорідності* див. пункт III.2.3).

Довести, що неоднорідне означення сильніше за однорідне: якщо ансамблі поліноміально нерозрізнювані в неоднорідному сенсі, то вони поліноміально нерозрізнювані і в однорідному сенсі. Як наслідок, генератор, псевдовипадковий при неоднорідному означенні нерозрізнюваності, є псевдовипадковим і при однорідному означенні.

2.3. Нехай поліноміальний імовірнісний алгоритм розв'язує задачу обчислення з ймовірністю помилки ϵ, використовуючи на входах довжини n випадкову послідовність довжини n^d. Припустимо тепер, що замість справжньої випадкової послідовності використовується отримана із випадкового паростка довжини n послідовність такої ж довжини, псевдовипадкова у сенсі неоднорідного означення із попередньої вправи. Довести, що для досить довгих входів ймовірність помилки може зрости щонайбільше на $1/n^c$, для довільної наперед заданої константи c.

2.4. Довести теорему 2.7.

2.5. Для генератора $G : \{0,1\}^n \to \{0,1\}^{l(n)}$ позначимо через $Z_{l(n)}$ випадкову величину $G(x)$ зі значеннями в $\{0,1\}^{l(n)}$, де x рівномірно розподілений на $\{0,1\}^n$. Припустимо, що ансамбль $\{Z_{l(n)}\}_{n\in\mathbb{N}}$ непередбачуваний, тобто, знаючи попередні біти послідовності $Z_{l(n)}$, черговий біт неможливо передбачити у сенсі означення 2.6. Довести, що тоді у такому ж сенсі, знаючи лише кінцевий відрізок бітів, неможливо вгадати попередній до них біт.

2.6. Довести теорему 2.8 в такий спосіб. Припустити, що G не витримує тестування наступного біту і виснувати звідси, що $B(x)$ можна ефективно отримати за $f(x)$ з ймовірністю, переважаючою $1/2 + 1/n^c$ для деякої константи c.

2.7. Сконструювати псевдовипадковий генератор, застосувавши загальну конструкцію пункту 2.3 до важкооборотної перестановки a) RSA, b) EXP.

2.8. Довести, що в позначеннях з опису BBS генератора,

$$x_i = x_0^{2^i \bmod \phi(m)} \bmod m,$$

де $\phi(m) = (p-1)(q-1)$.

2.9. Використовується криптосистема Блюма-Гольдвассер з таємним ключем $p = 19$ і $q = 23$.

a) Обчислити відкритий ключ і зашифрувати повідомлення **ТАК**. При генеруванні псевдовипадкової послідовності використати паросток 250.

b) Дешифрувати криптотекст (111101 111111, 73).

ЛІТЕРАТУРА

Фізичні способи отримання випадкових послідовностей описані у [137].

Деякі добре відомі генератори псевдовипадкових чисел з успіхом використовуються на практиці у алгоритмах Монте Карло (див. [24, § 2 розділу 8] та [32, розділ 3]). На жаль, ці генератори непридатні для криптографічних застосувань, оскільки не є криптографічно надійними. Прикладом може служити *лінійний генератор*, який на основі випадкового паростка $x_0 \in \mathbb{Z}_m$ породжує послідовність $x_0, x_1, x_2, \ldots$ згідно із рекурентним співвідношенням $x_i = (ax_{i-1} + b) \bmod m$. У [125] показано, що суперник, грунтуючись лише на кількох членах послідовності $x_0, x_1, x_2, \ldots$, може визначити параметри a, b, і m, і таким чином отримати будь-який член послідовності (див. також [10]). Як показано в [109], послідовність, що є розкладом у нескінченний двійковий дріб будь-якого алгебраїчного числа, на зразок $\sqrt{2} = 1,011010100\ldots$, також є криптографічно ненадійною.

Теореми 2.7 та 2.8 доведено в [150] та [79], відповідно. В останній роботі загальну конструкцію псевдовипадкового генератора було реалізовано для важкооборотної перестановки EXP. Доведення підсилення теореми 2.8, згаданого у зауваженні 2.9, можна знайти в [93, твердження 3.17].

Конструкцію псевдовипадкового генератора на основі довільної важкооборотної функції (не обов'язково перестановки) розроблено в [107, 103]. Таким чином, існування псевдовипадкових генераторів випливає з існування важкооборотних функцій. Обернене твердження встановлюється просто (деталі в [93]).

У [94, 117] показано як маючи генератор псевдовипадкових бітів, отримати *симетричну* криптосистему, надійність якої можна формально довести.

Розділ VII.

Протоколи

Під *протоколом* слід розуміти послідовність узгоджених приписів, згідно з якими відбувається обмін повідомленнями між *сторонами* або *учасниками* протоколу задля досягнення певної мети. Протоколи, яким присвячено цей розділ, є прикладами успішного застосування ідеології відкритого ключа до розв'язання різноманітних задач, пов'язаних із захистом інформації. Ці протоколи виглядають тим більш естетичними і ефектними, що самі задачі не можна було навіть поставити у рамках класичної криптографії.

§ 1. Обмін ключем

Традиційна симетрична схема захисту конфіденційності листування грунтується на наявності надійного каналу для обміну таємним ключем. Канал цей може бути набагато повільнішим, ніж канал для обміну повідомленнями, але безумовно він повинен бути захищеним від посягань суперника. У типовому класичному випадку такий канал реалізується з допомогою кур'єра, який доставляє ключ від одного користувача до іншого.

В асиметричних криптосистемах проблеми пересилання ключа не існує, адже таємний ключ є особистою власністю кожного абонента мережі, а відкритий ключ перебуває у відкритому доступі. Зазначимо однак, що з появою асиметричних криптосистем симетричні системи не вийшли зі вжитку з тої причини, що останні є набагато швидшими. Фактор же швидкості криптування/декриптування стає визначальним

при пересиланні великих обсягів інформації. Проте асиметричні криптосистеми відкривають нові можливості для обміну ключами при використанні криптосистем симетричних. Наприклад, практичним є пересилання ключа тим же каналом зв'язку, що й звичайних повідомлень, але зашифрованого за допомогою асиметричної криптосистеми. І хоча швидкодія криптосистеми з відкритим ключем нижча, для цієї мети вона достатня, адже ключ буде посилатися значно рідше, ніж звичайні повідомлення.

Нижче наводиться інше елегантне розв'язання проблеми, а саме, протокол *експоненційного обміну ключем*. Учасниками протоколу є наші давні знайомі — Аліса і Боб, які, спілкуючись через канал, що ймовірно прослуховується, хочуть домовитися про спільний таємний ключ.

- Аліса вибирає велике просте число p та первісний корінь g за модулем p, і відкрито, не роблячи з цього жодної таємниці, посилає p і g Бобові.
- Аліса вибирає випадкове число a в межах від 1 до $p-1$, а Боб — випадкове число b в тих же межах.
- Аліса обчислює $g^a \bmod p$ і посилає це значення Бобові, а Боб обчислює $g^b \bmod p$ і теж посилає Алісі.
- І Аліса, і Боб обчислюють одне і теж число

$$(g^b)^a \bmod p = (g^a)^b \bmod p = g^{ab} \bmod p,$$

яке і приймають в якості ключа.

Приклад 1.1. Нехай $p = 97$, а $g = 5$. Припустимо, що Аліса вибрала число $a = 12$, а Боб вибрав $b = 63$. Тоді Аліса посилає Бобові $5^{12} \bmod 97 = 42$, він їй $5^{63} \bmod 97 = 75$, і обоє обчислюють $75^{12} \bmod 97 = 42^{63} \bmod 97 = 21$.

Обчислювальна робота, передбачена описаним протоколом, багато в чому збігається з тою, що проводиться у криптосистемі ЕльГамала. Щоб уникнути повторення, ми посилаємось на обговорення питань ефективності в § V.5.

Зупинимось на питанні надійності протоколу. Суперник, який перехоплює всі повідомлення між Алісою та Бобом, стає власником чисел $g^a \bmod p$ та $g^b \bmod p$ і хоче отримати результат виконання протоколу — спільний ключ $g^{ab} \bmod p$. Таким чином, перед суперником постає така обчислювальна задача.

Розкриття експоненційного обміну ключем

Задано: $p, g, x, y \in \mathbb{N}$, де $1 < g < p-1$, $x = g^a \bmod p$, $y = g^b \bmod p$ для деяких a і b.

Обчислити: $z = g^{ab} \bmod p$.

Як бачимо, ця задача ідентична до задачі *Розкриття системи ЕльГамала — 2*, яка на сьогодні вважається важкою (див. обговорення в § V.5).

ВПРАВИ

1.1. Розширити протокол експоненційного обміну ключем для **a)** трьох, **b)** чотирьох, **c)** n учасників, які хочуть домовитись про спільний, один для всіх, ключ. В двох останніх випадках постаратися зробити якнайменшою кількість обмінів повідомленнями між учасниками.

ЛІТЕРАТУРА

Протокол експоненційного обміну ключем винайшли Діффі та Гелман в [20]. Криптосистема ЕльГамала була запропонована пізніше на основі ідеї цього протоколу.

§ 2. Цифровий підпис

Розглянемо таку ситуацію. Брокер від Київського акціонерного банку "Геркулес" вигідно придбав на Токійській фондовій біржі портфель цінних паперів, виставлених Нью-Йоркським Сітібанком. Оплата має бути переведена з Києва на Волл Стріт протягом години. Подібні фінансові операції проводяться через електронні засоби зв'язку. Це робить неможливим використання традиційних засобів засвідчення платіжних документів на зразок великої гербової печатки та підпису головного бухгалтера. Але як тоді банкові вберегтися від злодія-інтелектуала, який добре знається і на фінансах, і на електроніці, і може від імені брокера послати вимогу перевести гроші на власний підставний рахунок? Це завдання вирішується за допомогою протоколу *цифрового підпису*. Ми почнемо з конкретної реалізації такого протоколу на базі системи RSA.

2.1. Підпис у системі RSA. Нагадаємо, що в системі RSA кожен абонент X має пару ключів — загальновідомий відкритий (n_X, e_X) і таємний d_X, який знає лише X і ніхто інший. Таким чином, будь-хто може скористатися алгоритмом шифрування E_X абонента X, але

тільки він сам володіє алгоритмом дешифрування D_X. Важливим є виконання таких співвідношень для довільного повідомлення M:

$$D_X(E_X(M)) = E_X(D_X(M)) = M. \tag{1}$$

Ці співвідношення зводяться до рівностей $(M^{e_X})^{d_X} = (M^{d_X})^{e_X} = M$ в $\mathbb{Z}_{n_X}$, і виражають той факт, що шифруюче відображення E_X та дешифруюче D_X є взаємно оберненими.

Припустимо тепер, що Аліса хоче послати Бобові повідомлення M таким чином, щоб той був певен, що повідомлення справді послане Алісою, а не її суперницею Агнесою. Для цього пропонується такий протокол, в якому (E_A, D_A) та (E_B, D_B) — алгоритми шифрування та дешифрування Аліси та Боба.

- Аліса обчислює $C = E_B(D_A(M))$ і посилає C Бобові.
- Боб, отримавши C, обчислює $M = E_A(D_B(C))$.

Коректність протоколу зводиться до рівності

$$E_A(D_B(E_B(D_A(M)))) = M,$$

яка випливає із співвідношень (1).

Ефективність. Аліса та Боб використовують ефективні алгоритми шифрування та дешифрування криптосистеми RSA. Зауважимо, що Аліса використовує свій приватний алгоритм D_A та відомий всім алгоритм E_B. Те ж саме із Бобом — він використовує особистий алгоритм D_B і загальновідомий алгоритм E_A.

Конфіденційність. Під конфіденційністю цього протоколу ми розуміємо його надійність як звичайної криптосистеми для пересилання повідомлень. В такому розумінні конфіденційність є досить інтуїтивним фактом, що грунтується на надійності системи RSA. Перед суперницею Аліси Агнешкою, яка підслухала криптотекст C, виникає завдання визначити M із $E_B(D_A(M))$. Припустимо, що Агнешка знає навіть набагато більше, ніж їй слід, а саме Алісин приватний алгоритм D_A. Тоді визначення M для суперниці рівносильне визначенню $D_A(M)$. Але визначення $D_A(M)$ із $E_B(D_A(M))$ є нічим іншим, як задачею зламу RSA[1].

Достовірність. Під достовірністю ми розуміємо таку властивість протоколу підпису: Боб, і не тільки він, а будь-хто інший, може впевнитися, що відправником повідомлення є саме Аліса. Що стосується

[1] Щойно викладені міркування опираються на припущення, що ключі Аліси та Боба вибираються незалежно один від другого.

описаного вище протоколу, то Боб, який успішно розшифрував криптотекст C і прочитав повідомлення M, дійсно має вагомі підстави вважати, що воно послане саме Алісою. Справді, криптотекст має структуру $C = E_B(D_A(M))$, тому інтуїтивно переконливим є висновок, що особа, яка послала C, мала би знати алгоритм D_A.[2] Але алгоритм D_A відомий лише Алісі (якщо тільки Агнесі не вдалося зламати систему RSA).

2.2. Загальна схема. Описана в попередньому пункті криптосистема забезпечує як достовірність повідомлення, так і його конфіденційність. Є системи власне цифрового підпису, метою яких є забезпечення лише достовірності. Такі системи вкладаються у схему, що включає в себе наступні складові:

- *Ймовірнісний алгоритм генерування ключів.* Кожен абонент A отримує пару (K_A, K'_A), де K_A — відкритий, а K'_A — таємний ключі.
- *Алгоритм підписування* $SIGN$, який отримавши на вхід довільне повідомлення M та таємний ключ K'_A, продукує слово $S = SIGN(M, K'_A)$ в деякому алфавіті, яке називається *підписом* абонента A на повідомленні M. Коли A хоче послати комусь повідомлення M із запевненням, що воно відправлене саме ним, то посилає пару (M, S).
- *Алгоритм підтвердження підпису* $CHECK$, який є до послуг будь-кого, хто забажає перевірити, що підпис S повідомлення M належить саме власникові відкритого ключа K_A. Перевірка вважається успішною, якщо $CHECK(K_A, M, S) = 1$. Для будь-якого повідомлення M і для кожної пари ключів (K, K') має виконуватись співвідношення
$$CHECK(K, M, SIGN(M, K')) = 1.$$
Виконання цієї умови означає *коректність* системи цифрового підпису.

Всі алгоритми — генерування ключів, підписування та перевірки підпису — повинні бути ефективними.

Надійність такої системи підпису означає, що лише законний власник таємного ключа K' може для повідомлення M виробити такий підпис S, який пройшов би перевірку, тобто для відповідного відкритого ключа K виконувалась рівність $CHECK(K, M, S) = 1$. Якщо ж такий

[2]Цей висновок стає бездоганним, якщо припустити, що суперник/суперниця звідкись знає алгоритм D_B. Дивись також попередню примітку.

підпис S знаходить суперник, то кажуть, що він *підроблює* або *фальшує* підпис легального абонента на повідомленні M.

Загроза підроблення підпису суперником може бути різних ступенів:

- *екзистенційне фальшування* — існує повідомлення M, підпис якого суперник може підробити (хоча це повідомлення може для нього не мати жодного інтересу);
- *вибіркове фальшування* — суперник здатен підробити підпис деяких повідомлень за власним вибором;
- *універсальне фальшування* — суперник може підробити підпис будь-якого повідомлення, хоча й не знає таємного ключа;
- *повний злам системи підпису* — суперник спроможний визначити таємний ключ.

При фальшуванні підпису суперник виходить із наявної у нього інформації. Залежно від того, якою саме інформацією він володіє, розрізняють різні види атак суперника (нагадаємо, що алгоритми генерування ключів, підписування та перевірки вважаються відомими суперникові завжди):

- *атака за ключем* — суперник знає лише відкритий ключ K_A абонента A;
- *атака за відомим підписом* — суперник знає пару (M, S), де повідомлення M було вибране абонентом A і $S = SIGN(M, K'_A)$;
- *атака з вибором повідомлень* — суперник має можливість вибрати на власний розсуд певну кількість повідомлень $M_1, \ldots, M_k$ і для кожного M_i отримати підпис $S_i = SIGN(M_i, K')$. (Наприклад, абонент A працює нотаріусом і зобов'язаний засвідчувати власним підписом електронні копії показаних йому паперових документів.)

Описана у попередньому пункті система цифрового підпису на базі RSA є частковим випадком загальної схеми, запропонованої Діффі та Гелманом в [20]. За цією схемою кожну криптосистему з відкритим ключем можна перетворити в систему цифрового підпису наступним чином. Нехай E і D — алгоритми шифрування і дешифрування, K і K' — відкритий і таємний ключі, а M — довільне повідомлення. Тоді $SIGN(M, K') = D_{K'}(M)$ і

$$CHECK(K, M, S) = \begin{cases} 1, & \text{якщо } E_K(S) = M; \\ 0, & \text{інакше.} \end{cases}$$

2.3. Система цифрового підпису ЕльГамала. Як і криптосистеми для забезпечення конфіденційності повідомлень, системи цифрового

підпису допускають ймовірнісну модифікацію. У ймовірнісних системах підпису алгоритм $SIGN$ є не детермінованим, а ймовірнісним. Він продукує підпис S, що є випадковою величиною. У цьому і наступних пунктах 2.5 та 2.6 описуються деякі популярні серед практиків ймовірнісні системи цифрового підпису.

Першою ми розглянемо систему ЕльГамала [88]. Вона грунтується на тій же ідеї, що й криптосистема із § V.5.

Генерування ключів. Вибирають велике просте p, а також число g, $1 < g < p-1$, яке має в мультиплікативній групі $\mathbb{Z}_p^*$ великий порядок. В ідеальному випадку g є первісним коренем за модулем p. Числа p і g не є таємницею і перебувають в загальному користуванні. Кожен абонент вибирає собі випадкове число a у проміжку від 1 до $p-1$, і обчислює $h = g^a \bmod p$.

Відкритий ключ: p, g, h.

Таємний ключ: a.

Підписування. Аліса виробляє свій підпис S на повідомленні M таким чином. Вона

- вибирає випадкове число $r \in \mathbb{Z}_{p-1}^*$;
- обчислює $s_1 = g^r \bmod p$;
- обчислює $r' = r^{-1} \bmod (p-1)$;
- обчислює $s_2 = (M - as_1)r' \bmod (p-1)$;
- покладає $S = (s_1, s_2)$.

Підтвердження підпису.

- Боб перевіряє, чи $g^M \equiv h^{s_1} s_1^{s_2} \pmod p$.

Коректність. З правил обчислення r' і s_2 випливає, що $M \equiv as_1 + rs_2 \pmod{p-1}$. Звідси з використанням теореми Ойлера отримуємо

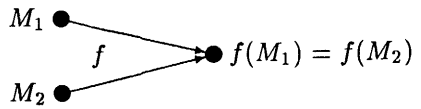
$$g^M = g^{as_1+rs_2+k(p-1)} \equiv (g^a)^{s_1}(g^r)^{s_2} \equiv h^{s_1}s_1^{s_2} \pmod p.$$

2.4. Коди достовірності. В цьому пункті ми ознайомимось із корисним засобом, який доцільно використовувати в будь-якій системі цифрового підпису.

Нехай функція f відображає повідомлення M довільної довжини у слово фіксованої довжини, скажімо, 128-бітове. Припустимо, що f обчислюється ефективно. Тоді така функція називається *вкорочуючою*.

Пара різних повідомлень M_1 та M_2, для яких виконується рівність $f(M_1) = f(M_2)$, називається *клешнею* або *колізією* функції f (див.

малюнок 1). Якщо для f неможливо за реалістичний час знайти колізію, то така функція вкорочення називається *безколізійною*.



Малюнок 1. Колізія вкорочуючої функції f.

Слід підкреслити, що оскільки розглядається функція f, яка є скінченним об'єктом, то слова "ефективно" і "за реалістичний час" вжиті вище не у формальному, а в інтуїтивному і суто практичному значенні.

Відзначимо важливу властивість безколізійної функції f — для більшості аргументів M досить великої довжини, за заданим образом $f(M)$ неможливо ефективно знайти таке M', що $f(M') = f(M)$ (див. вправу 2.3). Це не що інше, як властивість важкооборотності, що була предметом вивчення у § VI.1.

Образ $f(M)$ називають *вкороченням повідомлення* M. У системах цифрового підпису вигідно підписувати не саме повідомлення M, а його вкорочення $f(M)$ здійснене за допомогою обумовленої безколізійної вкорочуючої функції f, тобто варто покласти $S = SIGN(f(M), K')$. Виграш від цього очевидний: оскільки $f(M)$ має фіксовану довжину для як завгодно довгих повідомлень, то фіксованою буде й довжина підпису S. Достовірність такого підпису не знижується. Теоретично, у суперника з'являється зайвий шанс для фальшування — підглянутий підпис S повідомлення M він міг би використати як підпис іншого повідомлення M' такого, що $f(M') = f(M)$. Але це було б рівнозначним знаходженню суперником колізії для функції f, що вкрай малоймовірно з огляду на її безколізійність.

Варіант вкорочуючої функції з ключем називають *кодом достовірності*. Якщо f — вкорочуюча функція з ключем, то код достовірності повідомлення M рівний $f(M, K)$. Той факт, що $f(M, K)$ отримано справді з M може перевірити лише особа, яка знає ключ K. Код достовірності є криптографічним аналогом *контрольної суми*. І контрольну суму, і код достовірності долучають до повідомлення при його пересиланні, першу — щоб перевірити, чи повідомлення не було пошкоджене внаслідок природних перешкод у каналі зв'язку, а другий — щоб перевірити, чи повідомлення на шляху до адресата не було по-

вністю чи частково підмінене. Кожну вкорочуючу функцію f можна перетворити у функцію з ключем f', поклавши $f'(M,K) = f(MK)$.

Прикладом вкорочуючої функції може бути така функція f, яка конструюється на основі функції RSA. Якщо $|M| < \log_2 n$, то $f(M) = RSA(M)$. Для довших повідомлень $M = M_1 \dots M_{l-1} M_l$, розбитих на блоки відповідної довжини, функція задається рівністю $f(M) = RSA(f(M_1 \dots M_{l-1}) \oplus M_l)$.

На практиці використовуються швидші вкорочуючі функції, які вважаються безколізійними. Із найбільш популярних згадаємо MD5 та SHA. Перша спроектована Роном Райвестом, а друга — фахівцями з NIST та NSA (див. [137]). Всі такі функції вкорочують вхідний текст до 128 або й більше бітів. Зазначимо, що суттєво меншого вкорочення досягти неможливо, оскільки для довільної функції з множиною значень скажімо $\{0,1\}^{64}$, цілком реально знайти колізію *методом дня народження* (див. вправу 2.5).

2.5. Система Шнорра. В цьому пункті ми опишемо ймовірнісну систему цифрового підпису, запропоновану Клаусом Шнорром у [138].

Генерування ключів. Вибирають велике просте число p таке, що $p-1$ має досить великий простий дільник q (рекомендованими величинами є $p > 2^{512}$ і $q > 2^{140}$). Вибирають також число $h \neq 1$ таке, що $h^q \equiv 1 \pmod p$. Параметри p, q, h не становлять таємниці і є спільними для всіх абонентів мережі.

Аліса вибирає випадкове число a в діапазоні від 1 до $q-1$ і обчислює число $v = (h^a)^{-1} \bmod p$. Її ключі формуються так.

Відкритий ключ: v таке, що $h^a v \equiv 1 \pmod p$.

Таємний ключ: a.

Підписування. Алгоритм підписування використовує деяку вкорочуючу функцію f. Щоб виробити свій підпис S для повідомлення M, Аліса

- вибирає випадкове число r в діапазоні від 0 до $q-1$;
- обчислює $X = h^r \bmod p$;
- обчислює $s_1 = f(MX)$, де MX позначає результат злиття M і X в один текст;
- обчислює $s_2 = (r + as_1) \bmod q$;
- утворює $S = (s_1, s_2)$.

Підтвердження підпису. Отримавши повідомлення M із підписом $S = (s_1, s_2)$, Боб

- обчислює $Z = h^{s_2} v^{s_1} \bmod p$;

- перевіряє, чи $s_1 = f(MZ)$.

Коректність. Якщо компонента підпису s_2 утворена, як передбачено протоколом, то $Z = X$. Справді,

$$h^{s_2} v^{s_1} = h^{r+as_1+kq} v^{s_1} = h^r (h^q)^k (h^a v)^{s_1} \equiv h^r \pmod p.$$

Тому, якщо й s_1 отримано згідно з протоколом, то перевірка відбувається успішно.

2.6. DSA. У 1991 році NIST запропонував у якості стандарту цифрового підпису систему DSA (від Digital Signature Algorithm). Цей стандарт носить назву DSS (від Digital Signature Standard) і опублікований в [122].

Генерування ключів. Вибирають велике просте число p таке, що $p-1$ має досить великий простий дільник q. Стандарт вимагає, щоб $2^{512} < p < 2^{1024}$ і $q > 2^{160}$. Далі вибирають у групі $\mathbb{Z}_p^*$ довільний елемент h порядку q (див. вправу 2.6). Параметри p, q, h не становлять таємниці і є спільними для всіх абонентів мережі.

Аліса вибирає випадкове число a в діапазоні від 0 до $q-1$ і обчислює число $b = h^a \bmod p$. Її ключі формуються так.

Відкритий ключ: b таке, що $b = h^a \bmod p$.

Таємний ключ: a.

Підписування. Алгоритм підписування використовує вкорочуючу функцію f, в якості якої DSS пропонує функцію SHA [123] з довжиною вкорочення 160 бітів. Щоб виробити свій підпис S для повідомлення M, Аліса

- вибирає випадкове число r в діапазоні від 0 до $q-1$;
- обчислює $r' = r^{-1} \bmod q$;
- обчислює $s_1 = (h^r \bmod p) \bmod q$;
- обчислює $s_2 = (r'(f(M) + as_1)) \bmod q$;
- формує підпис $S = (s_1, s_2)$.

Підтвердження підпису. Отримавши повідомлення M із підписом $S = (s_1, s_2)$, Боб

- обчислює $s' = s_2^{-1} \bmod q$;
- обчислює $u_1 = (f(M)s') \bmod q$;
- обчислює $u_2 = (s_1 s') \bmod q$;
- обчислює $t = (h^{u_1} b^{u_2} \bmod p) \bmod q$;
- перевіряє рівність $t = s_1$.

Коректність. Припустимо, що підпис $S = (s_1, s_2)$ отримано згідно з алгоритмом підписування. Досить перевірити, що $h^r \equiv h^{u_1} b^{u_2} \pmod p$.

Оскільки $b = h^a \bmod p$, то остання конгруенція рівносильна такій: $h^r \equiv h^{u_1+au_2} (\mathrm{mod} p)$. Оскільки h має порядок q в $\mathbb{Z}_p^*$, то необхідно і досить перевірити, що $r \equiv u_1 + au_2 (\mathrm{mod} q)$ або, еквівалентно, $1 \equiv r'(u_1 + au_2) (\mathrm{mod} q)$. Підставивши у праву частину вирази для u_1 та u_2, отримаємо

$$r'(u_1 + au_2) \equiv r'(f(M)s' + as_1s') = r'(f(M) + as_1)s' \equiv s_2s' \equiv 1 (\mathrm{mod}\, q),$$

що завершує перевірку коректності.

ВПРАВИ

2.1. Бобова секретарка Агнеса носить (посилає через локальну мережу) документи йому на підпис. Підписування здійснюється в системі RSA: подавши Бобові M, Агнеса отримує $D_B(M)$. Документи бувають двох типів: текстові, які Боб з цікавості перечитує перед тим як підписати, і цифрові — технічні чи фінансові звіти, які складаються з самих чисел, і які Боб підписує відразу, лінуючись вникати в суть. Пояснити, як Агнеса може отримати підпис Боба на текстовому документі, що його Боб не підписав би ні за яких обставин.

2.2. ([130]) Аліса тримає в таємниці два великі прості числа p та q, і оприлюднює їх добуток n. В якості підпису повідомлення M їй служить якийсь із квадратних коренів із M за модулем n (тут не йдеться про конфіденційність, а лише про достовірність). Розробити атаку з вибором повідомлення, яка б приводила до повного зламу цієї системи підпису.

2.3. Обгрунтувати, що безколізійна вкорочуюча функція є важкооборотною. Розглянути таку конкретну ситуацію. Нехай $f : \{0,1\}^* \to \{0,1\}^{128}$ не є важкооборотною з тої причини, що деякий ймовірнісний алгоритм A на вході $f(x)$ для випадкового $x \in \{0,1\}^{130}$ швидко знаходить деяке x', причому $f(x') = f(x)$ з ймовірністю $1/2$ (не виключено, що $x' = x$). Довести, що для випадкового $x \in \{0,1\}^{130}$ пара x і $x' = A(f(x))$ утворює колізію функції f з імовірністю принаймні $1/8$.

2.4. (Парадокс із днем народження)

a) Яке математичне сподівання кількості тих людей серед вибраних випадково n чоловік, чий день народження випадає того ж дня, що й у Вас[3]? Розглянути випадки $n = 28, 365$.

b) Яке математичне сподівання кількості пар людей з однаковим днем народження серед вибраних випадковим чином n чоловік? Розглянути випадок $n = 28$.

[3]Якщо Ваш день народження 29 лютого, пропустіть цей пункт.

c) Нехай із сукупності з n різних предметів вибираються з поверненням $\alpha\sqrt{n} + 1$ предмет. Довести, що поміж вибраних предметів з імовірністю щонайменше $1 - e^{-\alpha^2/2}$ знайдуться хоча б два однакові.

d) Оцінити знизу ймовірність того, що серед 28 чоловік знайдуться двоє з однаковим днем народження.

2.5. (Метод дня народження)

a) Припустимо, що функція $f : \{0,1\}^* \to \{0,1\}^{64}$ володіє такою властивістю однорідності. Для будь-якого фіксованого $y \in \{0,1\}^{64}$ ймовірність того, що $f(x) = y$ для випадкового елемента $x \in \{0,1\}^k$, де $k \geq 64$, дорівнює 2^{-64}. Яке математичне сподівання кількості пар x і x' з однаковим образом $f(x) = f(x')$ серед 2^{33} елементів, випадково вибраних з $\{0,1\}^k$? Запропонувати ймовірнісну процедуру знаходження колізії для функції f.

Зауваження. Обчислення функції f від 2^{33} аргументів не є непосильним завданням для сучасної обчислювальної техніки за умови, що однократне обчислення f займає мало часу.

b) Нехай функція f така як у попередньому пункті, t — деякий фіксований текст довжини $k \geq 64$ і x — випадковий текст такої ж довжини. Нехай $T = \{ th : h \in \{0,1\}^{32} \}$ і $X = \{ xh : h \in \{0,1\}^{32} \}$. Чому дорівнює математичне сподівання кількості пар t' і x' таких, що $t' \in T$, $x' \in X$ і $f(t') = f(x')$?

c) ([151]) Нотаріус використовує систему цифрового підпису із вкорочуючою функцією $f : \{0,1\}^* \to \{0,1\}^{64}$. Пояснити, як зловмисник може з вагомою ймовірністю успіху отримати підпис нотаріуса на фальшивому документі t, давши нотаріусу на підпис "випадковий" безневинний документ x.

2.6. Нехай p та q — прості числа, і $q \mid p - 1$.

a) Припустимо, що g — первісний корінь за модулем p. Довести, що елемент $h = g^{(p-1)/q} \bmod p$ має порядок q в групі $\mathbb{Z}_p^*$.

b) Нехай g — випадковий елемент групи $\mathbb{Z}_p^*$, і h задається як у попередньому пункті. Довести, що з імовірністю $1 - 1/q$ порядок елемента h в $\mathbb{Z}_p^*$ дорівнює q.

ЛІТЕРАТУРА

Класифікація атак проти цифрового підпису і мір їх успіху запозичена з [131]. Системам цифрового підпису на базі важкооборотних функцій присвячена серія досліджень [100, 79, 120, 133]. В останній із цих робіт показано, що довільної важкооборотної функції досить, щоб сконструювати систему екзистенційно непідроблюваного підпису, стійку проти атаки з вибором повідомлень. Іншу техніку цифрового підпису запропоновано в [90].

§ 3. Підкидання монети по телефону

3.1. Попередне обговорення. Уявимо таку ситуацію. Національні футбольні збірні України та Бразилії вирішили провести одну товариську гру і їм належить домовитися, на чиєму полі вона відбудеться. Справедливо було б вирішити це питання жеребкуванням. Чи обов'язково з цією метою командам влаштовувати особисту зустріч своїх представників? Протокол *підкидання монети* дозволяє провести жеребкування на відстані за допомогою телефону (телеграфу, електронної пошти тощо).

Ідею протоколу можна зрозуміти на такому прикладі. Припустимо, що Аліна мешкає в Києві, а Боб у Ріо-де-Женейро. Нехай їм потрібно вирішити, хто до кого приїздить в гості. Вони домовляються спільними зусиллями згенерувати випадковий біт r, з однаковою ймовірністю рівний 1 або 0, з тим, що коли випаде $r = 0$, то Боб прилітає до Аліни, а коли $r = 1$, то Аліна вилітає в Ріо. Питання полягає в тому, як отримати r.

1-ий сценарій. Аліна підкидає монетку і результат — "чіт" ($r = 0$) чи "лишок" ($r = 1$) повідомляє Бобові телефоном. Таке вирішення проблеми дещо не влаштовує Боба — він не до кінця впевнений, що Аліна не схитрувала.

2-ий сценарій. І Аліна, і Боб підкидають кожен свою монетку. Нехай у Аліни випало $a \in \{0,1\}$, а у Боба $b \in \{0,1\}$. Вони покладають $r = a \oplus b$. Зрозуміло, що якщо a і b — випадкові біти, то r також. Більше того, навіть якщо хтось один з партнерів хитрує, але інший поводиться чесно, то r все одно набуває значення 0 та 1 рівноймовірно (довести!). Тому це добрий підхід до справи, але заковика в тому, що щойно сказане стосується лише випадку, коли a і b незалежні. В дійсності ж Аліна може спершу вислухати результат підкидання монети Бобом, а тоді у якості свого назвати той же біт, забезпечивши цим $r = 0$ і уникнувши довгих перельотів, яких вона недолюблює.

3-ій сценарій показує, як обійти цю проблему. Аліна підкидає монетку, записує біт a, і авіапоштою посилає його Бобові. Наступного дня, коли лист із штемпелем Київської головпошти вже в дорозі, але ще не дійшов за призначенням, Аліна та Боб зідзвонюються, Боб першим називає свій біт b, після чого Аліна називає свій. Цього разу вона не може схитрувати, тому що згодом Боб отримає біт a поштою і зможе виявити підміну.

3.2. Протокол на основі ймовірнісного криптування. В наступному протоколі реалізовано 3-ій сценарій, лише контрольне посилання біта a авіапоштою вирішується в режимі реального часу суто криптографічними засобами. Протокол використовує довільну надійну систему ймовірнісного криптування (див. § V.4). Через E і D позначимо алгоритми шифрування і дешифрування цієї системи, а через K і K' — відкритий і таємний ключі. Нагадаємо, що $E_K(0)$ та $E_K(1)$, результати шифрування однобітових повідомлень, є випадковими величинами, які неможливо ефективно розрізнити з імовірністю, суттєво більшою за $1/2$. Буде використовуватись ще одна властивість, якою володіють всі конкретні криптосистеми з відкритим ключем: за заданою парою (K, K') можна ефективно перевірити, чи вона справді породжена алгоритмом генерування ключів системи.

Щоб отримати біт r, який вони обоє визнають за випадковий, Аліна і Боб чинять так.

- Аліна вибирає свій біт a і пару ключів (K, K'). Після цього зашифровує a і результат $c = E_K(a)$ разом з ключем K посилає Бобові.
- Боб вибирає біт b і посилає його Аліні.
- Аліна посилає Бобові дешифруючий ключ K'.
- Боб перевіряє, що (K, K') справді є парою шифруючого та дешифруючого ключів, і обчислює $a = D_{K'}(c)$.
- Обоє обчислюють $r = a \oplus b$.

Як і у описаному вище 3-му сценарії, Аліна позбавлена можливості злукавити, бо посилає свій біт ще перед тим, як довідається Бобів. Боб теж не здатен хитрувати, бо отримує Алінин біт у зашифрованому вигляді і не може його відтворити без дешифруючого ключа K', звичайно за умови, що використовується надійна криптосистема.

Подивимось, як виглядає конкретне втілення поданого вище протоколу з використанням ймовірнісного криптування на основі розпізнавання квадратичності.

- Аліна вибирає досить великі прості числа p та q і обчислює їх добуток $n = pq$, після чого вибирає біт a. Далі вона вона генерує число $x \in \mathbb{J}_n$ із символом Якобі $\left(\frac{x}{n}\right) = 1$ так, що x є випадковим квадратичним лишком за модулем n, якщо $a = 0$, і випадковим псевдоквадратом, якщо $a = 1$.
- Аліна посилає Бобові x і n.
- Боб вибирає біт b і посилає його Аліні.
- Аліна відкриває Бобові прості p і q.

- Боб пересвідчується, що справді $pq = n$. Далі він застосовує алгоритм із пункту IV.4.3 і з його допомогою визначає, чи $x \in Q_n$, тим самим отримуючи біт a.
- Аліна і Боб обоє обчислюють $r = a \oplus b$.

3.3. Протокол на основі дволистої функції із секретом. Розглянемо ще один протокол підкидання монети. Він використовує *дволисту* важкооборотну функцію із секретом. Мається на увазі функція f з такими властивостями:

1) f обчислюється ефективно;
2) для кожного x існує єдине $x' \neq x$ таке, що $f(x) = f(x')$;
3) x' можна ефективно обчислити за x, лише якщо знати секретний ключ K.

(У деталях важкооборотні функції розглядались в § VI.1)

Прикладом дволистої важкооборотної функції з секретом є функція Рабіна піднесення до квадрату за модулем n, якщо звузити її область визначення до $\{1, 2, \ldots, (n-1)/2\}$. Нагадаємо, що $n = pq$ є тут цілим числом Блюма, тобто добутком двох простих з властивістю $p \equiv q \equiv 3 \pmod 4$.

Новий протокол підкидання монети влаштований так.

- Аліса вибирає функцію f і повідомляє про це Боба, втаївши секретний ключ K.
- Боб вибирає випадковий елемент x, обчислює $y = f(x)$ і посилає y Алісі.
- Аліса за допомогою секретного ключа K обчислює обидва прообрази x_1 та x_2 елемента y: $f(x_1) = f(x_2) = y$. Потім вибирає один із цих прообразів, скажімо x_i, і посилає його Бобові.
- Якщо $x_i \neq x$, то Боб знає обидва прообрази $\{x_i, x\} = \{x_1, x_2\}$ елемента y і посилає їх Алісі. Аліса і Боб покладають $r = 1$.
- Якщо $x_i = x$, то Боб визнає, що не знає другого прообразу. В цьому випадку Аліса і Боб покладають $r = 0$.
- Аліса посилає Бобові секретний ключ K, і Боб перевіряє, що y має справді два прообрази.

ВПРАВИ

3.1. Аліна і Боб живуть в одному місті. Реалізувати 3-ій сценарій підкидання монети по телефону (пункт 3.1), виходячи з припущення, що обоє мають під рукою телефонну книгу.

3.2. Розробити протокол підкидання монети, який би використовував ймовірнісну криптосистему на основі

a) RSA функції (пункт V.4.3);

b) довільного предикату з секретом (пункт VI.1.4).

3.3. Пояснити, чому протокол з пункту 3.3 є *справедливим*, тобто ні Аліса, ні Боб не здатні вплинути на розподіл біту r (за умови, що хоча б хтось один із них не хитрує, тобто притримується протоколу).

3.4. Нехай n є цілим Блюма. Довести, що звуження функції $SQUARE(x) = x^2 \bmod n$ на область $\{1, 2, \ldots, (n-1)/2\}$ справді володіє трьома властивостями дволистої важкооборотної функції з секретом, за умови важкості факторизації цілих Блюма.

3.5. a) Реалізувати протокол підкидання монети з пункту 3.3 на основі функції *SQUARE* з областю визначення $\{1, 2, \ldots, (n-1)/2\}$.

b) Довести, що у випадку $r = 1$ Боб здатен самостійно визначити секретний ключ p, q.

ЛІТЕРАТУРА

Задача підкидання монети на відстані була поставлена і розв'язана в [76].

§ 4. Гра в карти заочно

Аліса і Боб, знаходячись на відстані одне від одного, вирішили розважитись грою в покер. Хоча зв'язок між ними добре налагоджений, скажімо, за допомогою Інтернету, залишається проблема чесної роздачі карт. Однак і ця проблема вирішується за допомогою протоколу *заочної гри в карти*.

Описаний нижче протокол використовує довільну *комутативну* криптосистему. Алгоритми шифрування і дешифрування Аліси позначимо через E_A і D_A, а Боба — через E_B і D_B. Комутативність криптосистеми означає, що для будь-якого повідомлення M справедлива рівність

$$E_B(E_A(M)) = E_A(E_B(M)).$$

Прикладом такої системи є модифікація RSA, де модуль n учасники вибирають спільно, а шифруючі/дешифруючі експоненти e і d — таємно одне від другого. Для іншого варіанту комутативної криптосистеми модуль n вибирається простим. В обидвох випадках учасник X вибирає пару із шифруючого ключа e_X і дешифруючого d_X з властивістю $e_X d_X \equiv 1 \pmod{\phi(n)}$, і здійснює шифрування/дешифрування за формулами

$$E_X(M) = M^{e_X} \bmod n, \quad D_X(M) = M^{d_X} \bmod n.$$

Перейдемо до опису протоколу.

- Аліса і Боб досягають згоди про кодування карт словами $M_1, \ldots, M_{52}$, і домовляються, яка саме комутативна криптосистема буде використовуватись.
- Обидвоє таємно одне від другого вибирають собі шифруючий та дешифруючий ключі.
- Аліса зашифровує повідомлення $M_1, \ldots, M_{52}$, перемішує випадковим чином криптотексти $E_A(M_1), \ldots, E_A(M_{52})$ і посилає їх Бобові.
- Боб вибирає випадкові п'ять криптотекстів, і посилає їх назад Алісі. Це карти, якими буде грати Аліса.
  Коментар. Боб не може визначити карт Аліси, бо не знає її ключа.
- Із карт, що залишилися, Боб вибирає ще п'ять $E_A(M_{i_1}), \ldots, E_A(M_{i_5})$ для себе.
  Коментар. Боб вибрав карти у зашифрованому вигляді і тепер повинен довідатись, які це власне карти. Це він може зробити лише за допомогою Аліси, але повинен подбати, щоб при цьому його карти не відкрилися і їй.
- Боб зашифровує відібрані $E_A(M_{i_1}), \ldots, E_A(M_{i_5})$ за допомогою власного ключа і криптотексти
  $E_B(E_A(M_{i_1})) = E_A(E_B(M_{i_1})), \ldots, E_B(E_A(M_{i_5})) = E_A(E_B(M_{i_5}))$ посилає Алісі.
- Аліса дешифрує отримані криптотексти і повертає Бобові результат $E_B(M_{i_1}), \ldots, E_B(M_{i_5})$.
  Коментар. Аліса, яка не знає ключів Боба, не може звідси визначити його карти.
- Боб дешифрує надіслані Алісою $E_B(M_{i_1}), \ldots, E_B(M_{i_5})$ і отримує свою п'ятірку карт $M_{i_1}, \ldots, M_{i_5}$.
- В кінці гри Аліса і Боб обмінюються ключами і перевіряють, що ніхто з них не хитрував.

ВПРАВИ

4.1. Припустимо, що при роздачі карт Аліса і Боб використовують комутативну криптосистему, яка є котроюсь із двох згаданих модифікацій системи RSA. Аліса довіряє Бобові вибрати коди карт $M_1, \ldots, M_{52}$ в $\mathbb{Z}_n$. Перші чотири карти є тузами, і Боб "позначає" їх, вибравши коди так, що $\left(\frac{M_i}{n}\right) = -1$ для $i \leq 4$ і $\left(\frac{M_i}{n}\right) = 1$ для $i > 4$. Яким чином під час подальшого виконання протоколу роздачі карт Боб зможе розпізнавати тузи?

ЛІТЕРАТУРА

Протокол заочної гри в покер запропоновано в [59]. Криптоаналіз проведено в [115, 84].

§ 5. Розподіл таємниці

Аліса і Боб тримають у сейфі цінні папери, які є їхнім спільним надбанням. Сейф замикається на два замки. Один ключ зберігається в Аліси, а інший у Боба. Завдяки цьому Аліса і Боб можуть розпоряджатися цінностями лише за обопільною згодою.

Якщо цінності є у спільному володінні Аліси, Боба та Вітольда, і будь-яке рішення приймається більшістю голосів, то потрібен сейф складнішої конструкції. Замок повинен мати три отвори для трьох різних ключів, причому він повинен відкриватися будь-якою парою ключів, і не повинен відкриватися лише одним ключем.

Подібна ідея розподілу ключа між кількома особами покладена в основу протоколу *розподілу секрету*. Нехай натуральне число s є цінною секретною інформацією (номер рахунку у швейцарському банку, код команди на запуск балістичної ракети тощо). Завданням протоколу є так подрібнити секрет s на частини, по одній для кожного із n учасників, щоб будь-які k учасників могли відновити s, поєднавши свої частинки, але щоб ніяка група з $k-1$ учасника цього зробити не могла. Формально йдеться про процедуру, яка б співставляла числу s послідовність чисел $s_1, \ldots, s_n$ і при цьому виконувались такі дві умови:

1) s можна ефективно отримати з довільної k-елементної підпослідовності $s_{i_1}, \ldots, s_{i_k}$;
2) s в принципі не можна отримати ні з якої $(k-1)$-елементної послідовності.

Ми опишемо протокол, запропонований Аді Шаміром в [140].

Вибирають досить велике просте число p і розглядають s як елемент поля $\mathbb{Z}_p$. Покладають $a_0 = s$, і вибирають в $\mathbb{Z}_p$ випадковим чином числа $a_1, \ldots, a_{k-1}$. Нехай $f(x) = \sum_{0 \leq i < k} a_i x^i$ — многочлен від змінної x з коефіцієнтами в $\mathbb{Z}_p$. i-ий учасник протоколу, де $1 \leq i \leq n$, отримує значення $s_i = f(i)$.

Якщо відомо довільні k значень $f(i_1), \ldots, f(i_k)$, то многочлен f можна реконструювати за інтерполяційною формулою Лагранжа:

$$f(x) = \sum_{l=1}^{k} f(i_l) \prod_{j \neq l} \frac{x - i_j}{i_l - i_j}.$$

Після цього легко знаходиться секрет $s = a_0 = f(0)$.

Знання лише $k-1$ значення $f(i_1), \ldots, f(i_{k-1})$ не дає *жодної* інформації про секрет. Справді, за тією ж інтерполяційною формулою Лагранжа, для будь-якого s' існує поліном f із заданими значеннями в $i_1, \ldots, i_{k-1}$ і такий, що $f(0) = s'$.

Таким чином, наведений протокол є ще одним прикладом криптосистеми, надійної у теоретико-інформаційному сенсі.

ВПРАВИ

5.1. Розділити секрет $s = 13$ між $n = 3$ особами з кворумом $k = 2$, вибравши $p = 17$.

5.2. Довести, що многочлен степеня $k-1$ від однієї змінної з коефіцієнтами у полі F однозначно визначається своїми значеннями в різних k точках поля F за інтерполяційною формулою Лагранжа.

§ 6. Доведення без розголошення

Якось Аліса і Боб знічев'я блукали в мережі Інтернет і ненароком натрапили на захищену базу даних. За якийсь час Боб зламав систему захисту і дуже хотів похизуватися цим перед Алісою. Водночас, як людина відповідальна, Боб не збирався відкривати Алісі чужих таємниць, бо напевне знав, що вона відразу обдзвонить всіх своїх найкращих подруг. Наміри Боба здаються суперечливими і нездійсненними, однак їх можна реалізувати, скориставшись протоколом *доведення без розголошення*. Внаслідок виконання цього протоколу Боб за допомогою секретної інформації переконує Алісу у справедливості певного твердження, але Аліса при цьому не отримує цієї секретної інформації, більше того, у певному розумінні не отримує ніякої інформації взагалі. Таким чином, Аліса, хоча сама впевнюється, що твердження має місце, не здатна переконати в цьому ніяку третю особу.

6.1. Доведення квадратичності. В якості першого прикладу ми розглянемо доведення факту, що деяке число $x \in \mathbb{Z}_n^*$ є квадратичним лишком за модулем n. Як зазначалося в § IV.4, задача розпізнавання

квадратичних лишків є важкою, якщо тільки не є відомим розклад модуля n на прості множники. Тому Аліса за парою (x, n) неспроможна (за поліноміальний від $\log n$ час) визначити, чи справді x є квадратичним лишком.

Припустимо, що на відміну від Аліси, Боб знає якийсь квадратний корінь з x за модулем n. Позначимо цей корінь через y: $y^2 \equiv x \pmod{n}$. Нас не цікавить яким саме чином Боб добув корінь з x. Найправдоподібнішим поясненням таких виняткових обчислювальних здібностей може бути те, що Боб або власноручно вибрав n і знає його розклад на множники, або спершу вибрав y, а вже потім обчислив $x = y^2 \bmod n$.

Найпростіший спосіб, у який Боб може переконати Алісу, що $x \in \mathcal{Q}_n$, це просто показати їй y. А наступний протокол дозволяє переконати Алісу не лише без пред'явлення їй коренів, але й без надання їй жодної інформації, яку Аліса не змогла б здобути самотужки.

Вхід: (x, n), де $x \in \mathbb{Z}_n^*$ і існує y таке, що $y^2 \equiv x \pmod{n}$.

- Боб вибирає випадковий елемент v в $\mathbb{Z}_n^*$, підносить його до квадрату і результат $u = v^2 \bmod n$ посилає Алісі.
- Аліса посилає Бобові випадковий біт $b \in \{0, 1\}$.
- Боб посилає Алісі число

$$w = \begin{cases} v, & \text{якщо } b = 0; \\ vy, & \text{якщо } b = 1. \end{cases}$$

- Якщо $b = 0$, то Аліса перевіряє, чи справді $u \equiv w^2 \pmod{n}$, а якщо $b = 1$, то перевіряє, чи справді $xu \equiv w^2 \pmod{n}$. Якщо результат перевірки негативний, то Аліса припиняє діалог і звинувачує Боба в ошуканстві.

Все вищеописане повторюється $m = \lceil \log n \rceil$ разів, щоразу із незалежним вибором випадкових величин. Якщо результат перевірки у кожному циклі був успішним, то Бобові вдається переконати Алісу, що $x \in \mathcal{Q}_n$.

Перелічимо три основні властивості цього протоколу.

1) *Повнота.* Якщо справді $x \in \mathcal{Q}_n$, а Боб справді знає квадратний корінь y і діє згідно з протоколом, то він переконує Алісу з імовірністю 1. Цю властивість легко перевірити, безпосередньо прослідкувавши за перебігом протоколу.

2) *Обгрунтованість.* Якщо ж $x \notin \mathcal{Q}_n$, то як би винахідливо Боб не діяв, тобто які б хитрі повідомлення u і w не посилав Алісі, вона викриє його нещирість з імовірністю $1/2$ в кожному із циклів, і з ймовірністю $1-2^{-m}$ хоча б в одному з m циклів. Таким чином, Боб може переконати

Алісу у справедливості хибного твердження з імовірністю щонайбільше $1/n$, що є експоненційно малою величиною в порівнянні з довжиною входу.

Щоб довести цю властивість, розглянемо два випадки.

Випадок 1: Перше повідомлення Боба Алісі u не є квадратичним лишком за модулем n. Тоді Боба буде викрито при $b = 0$, що трапиться з імовірністю $1/2$.

Випадок 2: Повідомлення u є квадратичним лишком за модулем n. Тоді xu не є квадратичним лишком і Боба буде викрито при $b = 1$, що теж трапиться з імовірністю $1/2$.

3) *Відсутність розголошення.* Якщо справді $x \in \mathcal{Q}_n$, то Аліса не отримує від Боба ніякої інформації. Це твердження досить переконливе інтуїтивно. Справді, якщо Аліса вибирає біт $b = 0$, то отримує випадковий квадратичний лишок u і корінь з нього w. Якщо ж $b = 1$, то Алісі дістається квадратний корінь w з xu, причому xu також є випадковим квадратичним лишком за модулем n (див. пункт **а** вправи III.3.8).

Для того, щоб формалізувати це спостереження, позначимо через $VIEW_{x,n}$ всю інформацію, яку сприймає Аліса упродовж одного циклу протоколу на вході (x, n), а саме, $VIEW_{x,n} = \langle u, b, w\rangle$. Аліса спостерігає кожну конкретну трійку $\langle u, b, w\rangle$ з певною ймовірністю, тобто $VIEW_{x,n}$ є випадковою величиною.

Для довільного ймовірнісного алгоритму M результат його роботи на вході (x, n) позначимо через $M(x, n)$. Слід звернути увагу на те, що $M(x, n)$ є випадковою величиною, значення якої залежить від випадкової послідовності алгоритму.

Абсолютно строго властивість відсутності розголошення можна сформулювати так: *існує поліноміальний ймовірнісний алгоритм M такий, що при $x \in \mathcal{Q}_n$ випадкові величини $M(x, n)$ і $VIEW_{x,n}$ однаково розподілені.* Таким чином, $VIEW_{x,n}$ не містить для Аліси ніякої нової інформації в тому сенсі, що Аліса могла б отримати все абсолютно те ж саме, запустивши алгоритм M на вході (x, n).

Нам залишається описати алгоритм M, який симулює діалог Аліси і Боба на входах (x, n) при $x \in \mathcal{Q}_n$. Спочатку слід вибрати випадковий елемент $w \in \mathbb{Z}_n^*$ і випадковий біт $b \in \{0, 1\}$. Якщо $b = 0$, то покласти $u = w^2 \bmod n$. Якщо $b = 1$, то покласти $u = w^2 x^{-1} \bmod n$. Трійку $\langle u, b, w\rangle$ подати на вихід.

Насправді наведений протокол володіє властивістю відсутності розголошення у значно сильнішій формі. Зауважимо, що випадкова величина $VIEW_{x,n}$ залежить від дій Аліси. Ми виходили з того, що Аліса

притримується протоколу. Але ж вона може хитрувати в надії вивідати у Боба чимбільше інформації. Наприклад, Аліса може посилати Бобові біт b, вибраний не рівноймовірно, а за якимось іншим розподілом. З врахуванням цієї можливості, властивість відсутності розголошення розглянутого протоколу можна сформулювати у підсиленому варіанті. Нехай Аліса вибирає своє повідомлення b як результат роботи деякого поліноміального ймовірнісного алгоритму A на вході $\langle x, n, u\rangle$ (це вся інформація, доступна Алісі на момент вибору). Тоді існує поліноміальний ймовірнісний алгоритм M_A такий, що при $x \in \mathcal{Q}_n$ випадкові величини $M_A(x, n)$ і $VIEW_{x,n}$ однаково розподілені.

Протокол з такою властивістю відсутності розголошення називають *досконалим* доведенням без розголошення.

6.2. Доведення неквадратичності. Тепер розглянемо доведення без розголошення того факту, що $x \in \mathbb{Z}_n^*$ *не є* квадратичним лишком за модулем n. Зразу обмежимось випадком, коли $\left(\frac{x}{n}\right) = 1$, тобто $x \in \mathbb{J}_n$, бо інакше доведення давалося б простим обчисленням символу Якобі (див. пункт 5 твердження IV.1.7). Отже Боб, який вміє розпізнавати квадратичність за модулем n (наприклад, знаючи розклад n на прості множники), хоче переконати Алісу, що x не є квадратичним лишком, тобто належить множині псевдоквадратів $\widetilde{\mathcal{Q}}_n$.

Вхід: (x, n), де $x \in \widetilde{\mathcal{Q}}_n$.

- Аліса вибирає випадковий елемент v в $\mathbb{Z}_n^*$ і випадковий біт $b \in \{0, 1\}$. Далі Аліса обчислює
$$w = \begin{cases} v^2 \bmod n, & \text{якщо } b = 0, \\ xv^2, & \text{якщо } b = 1, \end{cases}$$
і посилає w Бобові.
- Боб визначає, чи w є квадратичним лишком за модулем n і посилає Алісі у відповідь біт
$$b' = \begin{cases} 0, & \text{якщо } w \in \mathcal{Q}_n, \\ 1, & \text{якщо } w \notin \mathcal{Q}_n. \end{cases}$$
- Аліса перевіряє, чи $b = b'$. Якщо це не так, Аліса припиняє спілкування з Бобом і оголошує йому недовіру.

Все повторюється $m = \lceil \log n \rceil$ разів, щоразу із незалежним вибором випадкових величин. Якщо результат перевірки у кожному циклі був успішним, то Бобові вдається переконати Алісу, що $x \notin \mathcal{Q}_n$.

Подібно до попереднього, цей протокол має такі властивості.

Повнота. Якщо справді $x \in \widetilde{Q}_n$, то v^2 є квадратичним лишком за модулем n, а xv^2 ні (див. вправу IV.1.12). Тому Боб може визначити біт b за повідомленням Аліси w. Отже, у цьому випадку Боб переконує Алісу з імовірністю 1.

Обгрунтованість. Якщо $x \in Q_n$, то w є випадковим квадратичним лишком за модулем n. Більше того, w і b є незалежними випадковими величинами (довести!). Тому як би Боб не хитрував, отримавши w, він може вгадати b з ймовірністю лише $1/2$. Щоб переконати Алісу, йому слід це зробити у кожному циклі, що може вдатися з імовірністю лише $2^{-m} < 1/n$.

Відсутність розголошення для цього протоколу вдається довести у трішки слабшій формі, ніж для попереднього. А саме, існує поліноміальний ймовірнісний алгоритм M такий, що при $x \in \widetilde{Q}_n$ випадкові величини $M(x,n)$ і $VIEW_{x,n} = \langle v, b, w, b' \rangle$ *статистично близькі*, тобто

$$\sum_h |\mathbf{P}\left[VIEW_{x,n} = h\right] - \mathbf{P}\left[M(x,n) = h\right]| < \frac{1}{(\log n)^c}$$

для будь-якої константи c при досить великих значеннях n. Сумування проводиться за всіма можливими значеннями h випадкових величин $M(x,n)$ і $VIEW_{x,n}$.

Протокол з такою властивістю відсутності розголошення називають *майже досконалим* або *статистичним* доведенням без розголошення.

Тема доведень без розголошення буде продовжена в § VIII.2.

ВПРАВИ

6.1. Для протоколу доведення квадратичності перевірити, що при $x \in Q_n$ випадкові величини $M(x,n)$ і $VIEW_{x,n}$ мають однаковий розподіл.

6.2. ([98]) Граф G на множині вершин $V = \{1, \ldots, n\}$ задається множиною ребер E. Формально, ребро — це множина, яка складається з двох вершин. Кажуть, що ребро $e = \{u, v\}$ з'єднує вершини u і v. Графи $G_1(V, E_1)$ і $G_2(V, E_2)$ називаються ізоморфними, якщо існує перестановка вершин $f : V \to V$ така, що $\{u, v\} \in E_1$ тоді і лише тоді, коли $\{f(u), f(v)\} \in E_2$.

Розробити доведення без розголошення того, що два задані графи $G_1(V, E_1)$ та $G_2(V, E_2)$ **a)** ізоморфні, **b)** неізоморфні.

ЛІТЕРАТУРА

Концепція доведення без розголошення з'явилась в [97] і була розвинута в [98].

§ 7. Ідентифікація

Протокол *ідентифікації* дозволяє абонентові засвідчити власну особу перед іншим абонентом, щоб вступити з ним в певні стосунки, або перед обчислювальною, комунікаційною чи фінансовою системою, щоб отримати до неї доступ. Для прикладу, в *інтелектуальних картках* реалізуються ті чи інші протоколи ідентифікації.

Загальновідомим розв'язком задачі ідентифікації є використання гасла, про що вже йшлося у пункті VI.1.2. Слабким місцем такого простого вирішення проблеми є те, що суперник може підслухати гасло. Нижче наводяться інші протоколи ідентифікації, які позбавлені цього недоліку.

7.1. Ідентифікація за допомогою симетричної криптосистеми. Аліса і Боб домовляються про використання криптосистеми $\langle E, D \rangle$ з таємним ключем K. Припустимо, що Боб вийшов на зв'язок з Алісою, але вона хоче наперед впевнитись, що має справу саме з ним. Для цього

- Аліса вибирає випадкове слово M досить великої довжини і посилає його Бобові.
- Боб зашифровує M і посилає Алісі криптотекст $C = E_K(M)$.
- Аліса перевіряє, чи справді $D_K(C) = M$.

Зрозуміло, що намагання суперника видати себе за Боба можуть претендувати на успіх лише за умови зламу ним криптосистеми $\langle E, D \rangle$. Описаний протокол називають протоколом *виклику-і-відгуку.*

7.2. Ідентифікація на підставі цифрового підпису. Задачу ідентифікації можна трактувати як частковий випадок задачі цифрового підпису в тому плані, що кожен протокол цифрового підпису можна використати для ідентифікації. Аліса посилає Бобові якесь випадкове повідомлення і просить його підписати. Отримавши у відповідь підпис і успішно перевіривши цього, Аліса може бути впевнена, що має справу саме з Бобом — за умови неможливості універсального фальшування підпису.

Зазначимо, що протокол підпису, який використовується для ідентифікації, має бути стійким проти атаки з вибором повідомлення. Інакше суперник, який кілька разів попросив Боба ідентифікувати себе, зміг би згодом це робити замість Боба.

7.3. Ідентифікація як доведення без розголошення. Доведення без розголошення відкривають принципово нові можливості проведен-

ня ідентифікації. Кожен абонент оприлюднює деяке твердження, доведення якого відоме лише цьому абонентові. Наприклад, Боб вибирає таємно від всіх два великі прості числа p і q, а також випадкове число $y \in \mathbb{Z}_n^*$, де $n = pq$, обчислює $x = y^2 \bmod n$ і оголошує, що x є квадратичним лишком за модулем n. Коли Аліса хоче пересвідчитись, що має справу саме з Бобом, вона просить співрозмовника довести Бобове твердження, для чого використовується протокол із пункту 6.1.

Боб, знаючи y, переконує Алісу в істинності твердження з імовірністю 1. Самозванець спроможний на це лише з імовірністю 2^{-m}, де m — кількість циклів. Більше того, навіть прослухавши сеанс ідентифікації Боба Алісою, суперник не здатен повторити його замість Боба — доведення ж бо було без розголошення.

ВПРАВИ

7.1. Вибрати конкретні a) симетричну криптосистему, b) систему цифрового підпису, c) протокол доведення без розголошення, і розробити на їх основі протоколи ідентифікації.

ЛІТЕРАТУРА

При практичній реалізації протоколів ідентифікації найбільше часу йде не на обчислення (левову частку яких, пов'язану з вибором таємних і відкритих параметрів, можна провести завчасу), а на зв'язок між користувачами. Тобто найдорожчою операцією є обмін повідомленнями. Тому описаний в пункті 7.3 протокол ідентифікації покликаний лише проілюструвати загальний принцип, але не є практичним з огляду на те, що він використовує протокол доведення без розголошення з пункту 6.1, який потребує великої кількості циклів m. Питання про можливість зменшення загальної кількості обмінів повідомленнями в доведеннях без розголошення досліджене в [95, 124]. Практичні протоколи ідентифікації, що використовують ідеологію доведень без розголошення, запропоновані в [92, 91].

Про методи ідентифікації, що грунтуються на інших ідеях, йдеться в [52, 51, 137, 43].

Розділ VIII.

Питання $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ і криптографія

§ 1. Клас $\mathcal{NP}$

У цьому параграфі ми будемо розглядати в основному лише задачі розпізнавання. Ми не будемо конкретизувати, який саме алфавіт використовується для формулювання тої чи іншої задачі. Без втрати загальності можна вважати, що все кодується у двійковому алфавіті. Окрім того, як і у розділі III ми будемо використовувати допоміжний символ #.

1.1. Недетерміновані обчислення. Клас мов, які розпізнаються поліноміальними детермінованими алгоритмами, позначають через $\mathcal{P}$. Таким чином, $\mathcal{P} = \{L : \text{мова } L \text{ розпізнається деяким поліноміальним алгоритмом}\}$. У дещо іншій термінології $\mathcal{P}$ вважається класом задач розпізнавання, які розв'язуються за поліноміальний час. Дамо тепер означення ширшого класу, який має назву $\mathcal{NP}$.

Означення 1.1. Мова L (або задача її розпізнавання) належить класові $\mathcal{NP}$, якщо існують мова $\widehat{L} \in \mathcal{P}$ і константа $c \in \mathbb{N}$ такі, що для будь-якого слова w виконується така умова:

$$w \in L \text{ тоді і лише тоді, коли } w\#u \in \widehat{L}$$
$$\text{для деякого слова } u \text{ довжини } |u| \le |w|^c.$$

Якщо L входить в клас $\mathcal{NP}$, то кажуть, що L розпізнається *недетермінованим* поліноміальним алгоритмом. На вході w недетермінований алгоритм має можливість *вгадати* деяке слово u і після цього працює як звичайний детермінований алгоритм, використовуючи як w, так і

u. За означенням, недетермінований алгоритм розпізнає мову L, якщо виконуються такі дві умови:

1) якщо $w \in L$, то на вході w алгоритм подає на вихід 1 хоча б для одного здогаду u;
2) якщо $w \notin L$, то на вході w для будь-якого u алгоритм подає на вихід 0.

Слід звернути увагу на такий нюанс. Якщо мова $L \subset \{0,1\}^*$ розпізнається недетермінованим алгоритмом, то звідси не випливає автоматично те саме для її доповнення $\{0,1\}^* \setminus L$ — умови 1 і 2 не симетричні.

Зазначимо, що поняття недетермінованого обчислення не претендує на те, щоб бути реалістичною моделлю. Однак введення цього поняття має не лише теоретичну, а й практичну користь, про що йдеться нижче.

1.2. Приклади задач з класу $\mathcal{NP}$.

1) *Задано:* $w \in \mathbb{N}$.
Визначити: чи w складене?
Якщо вхід w є складеним числом, то здогадом u може служити будь-який простий дільник u числа w. Вгадавши u, алгоритм перевіряє, чи u справді ділить w, і якщо так, то видає 1, якщо ж ні, то 0.
2) *Задано:* x, n.
Визначити: чи $x \in \mathcal{Q}_n$?
Для входу $w = (x, n)$ здогадом може служити довільне u, яке є квадратним коренем з x за модулем n, тобто $x \equiv u^2 \pmod n$.

В пункті III.1.1 і прикладі III.4.1 було пояснено, як задачі обчислення і пошуку можна формулювати у вигляді задач розпізнавання. Задачі розпізнавання, що відповідають факторизації, добуванню квадратного кореня і дискретному логарифмуванню, теж входять у клас $\mathcal{NP}$.

3) *Задано:* n, s.
Визначити: чи існує u таке, що $1 < u \le s$, $u \neq n$ і $u \mid n$?
4) *Задано:* x, n, s.
Визначити: чи $x \in \mathcal{Q}_n$ і чи існує u таке, що $1 < u \le s$, $x \equiv u^2 \pmod n$?
5) *Задано:* x, g, n, s.
Визначити: чи існує u таке, що $1 < u \le s$, $x \equiv g^u \pmod n$?

1.3. Питання $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$. Банальним наслідком означень є включення $\mathcal{P} \subseteq \mathcal{NP}$. Питання, яка з альтернатив має місце — $\mathcal{P}$ є власним підкласом $\mathcal{NP}$, чи $\mathcal{P} = \mathcal{NP}$ — є відкритим. Це одна із центральних задач не лише теорії складності, а й сучасної математики взагалі. Загальноприйнятою гіпотезою є нерівність $\mathcal{P} \neq \mathcal{NP}$, однак на сьогодні доведення не існує.

Із укладеного в попередньому пункті списку деяких членів класу $\mathcal{NP}$ можна зрозуміти, що питання має безпосереднє відношення до криптографії. Упродовж нашого курсу ми бачили, що більшість практичних криптосистем та протоколів є надійними не в абсолютному, а в обчислювальному відношенні, тобто за умови, що такі задачі як факторизація, добування коренів, логарифмування тощо є важкими. Якщо ж $\mathcal{NP} = \mathcal{P}$, то перелічені в попередньому пункті задачі розпізнавання 3–5 розв'язуються за поліноміальний час. Відтак криптосистеми, надійність яких грунтується на важкості відповідних задач обчислення, могли би бути зламані за поліноміальний час. Взагалі, для природної формалізації понять симетричних та асиметричних криптосистем, задача зламування цих криптосистем є поліноміально еквівалентною до деякої задачі в класі $\mathcal{NP}$. Втім, практики не повинні брати близько до серця ці зауваження, поки серед теоретиків побутує впевненість, що $\mathcal{P} \neq \mathcal{NP}$.

1.4. $\mathcal{NP}$-повнота. Клас $\mathcal{NP}$ має визначну структурну властивість, яка особливо ефектно виглядає в контексті питання $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$.

Означення 1.2. Мова $L \in \mathcal{NP}$ (або задача її розпізнавання) називається *$\mathcal{NP}$-повною*, якщо будь-яка мова з класу $\mathcal{NP}$ поліноміально зводиться до L.

Принципове значення має

Теорема Кука-Левіна. *Існують $\mathcal{NP}$-повні мови.*

Власне, набагато більше від самого факту існування $\mathcal{NP}$-повних задач важить та обставина, що багато природних задач, які реально виникають у програмістській практиці, виявляються $\mathcal{NP}$-повними.

Твердження 1.3. *Якщо L є $\mathcal{NP}$-повною мовою, то $L \in \mathcal{P}$ тоді і лише тоді, коли $\mathcal{P} = \mathcal{NP}$.*

Доведення. В одну сторону твердження є очевидним. В іншу сторону воно є простим наслідком пункту 2 теореми III.4.2. ∎

У твердженні 1.3 полягає практична цінність концепції $\mathcal{NP}$-повноти. Якщо програмістові доручено розробити ефективний алгоритм розпізнавання мови L, в результаті кількох безуспішних спроб він

може запідозрити, що поставлена перед ним задача є $\mathcal{NP}$-повною. Тоді він може попросити фахівця з теорії складності довести $\mathcal{NP}$-повноту мови L. Якщо це буде зроблено, то програміст матиме право або відмовитись від задачі, або вимагати її послаблення, якщо лише він не збирається спростовувати гіпотезу $\mathcal{NP} \neq \mathcal{P}$.

1.5. Приклади $\mathcal{NP}$-повних задач.

1) *Задано:* функціональну схему S в базі $\{\vee, \oplus, 1\}$ із вхідними змінними $x_1, \ldots, x_m$ (див. пункт III.2.3).
Визначити: чи існують значення змінних, для яких результатом обчислення за схемою є 1?
2) *Задано:* граф $G(V, E)$, де $V = \{1, \ldots, n\}$ — множина вершин, а E — множина ребер. Якщо $\{u, v\} \in E$, то кажуть, що вершини u і v з'єднані ребром.
Визначити: чи можна вершини графа G розмалювати трьома кольорами так, щоб жодне ребро не з'єднувало вершини однакового кольору? Формально: чи існує відображення $\phi : V \to \{1, 2, 3\}$ з властивістю $\phi(u) \neq \phi(v)$ для кожного ребра $\{u, v\} \in E$?

Ще одним прикладом є задача 4 з пункту 1.2. Зазначимо, що задачі 1 і 2 з пункту 1.2 не вважаються $\mathcal{NP}$-повними. Для першої з цих задач інтуїція фахівців цілком зрозуміла — для жодної з відомих $\mathcal{NP}$-повних задач невідомо таких ефективних детермінованих чи ймовірнісних алгоритмів, як тести простоти, що розглядались нами в § IV.2.

Зауваження 1.4. Згідно з означенням 1.2, задача L з класу $\mathcal{NP}$ є $\mathcal{NP}$-повною, якщо кожна задача з $\mathcal{NP}$ поліноміально зводиться до L в сенсі пункту III.4.2. Таку звідність в теорії $\mathcal{NP}$-повноти називають *звідністю за Куком*. Розглянемо інше поняття звідності, так звану *звідність за Карпом*. Мова $L_1 \subseteq \{0,1\}^*$ зводиться до мови $L_2 \subseteq \{0,1\}^*$ за Карпом, якщо існує така поліноміально обчислювана функція f, що $w \in L_1$ тоді і лише тоді, коли $f(w) \in L_2$, для кожного $w \in \{0,1\}^*$. Так от, всі відомі $\mathcal{NP}$-повні задачі залишаються такими і відносно звідності за Карпом.

ВПРАВИ

1.1. Довести, що задачі розпізнавання 3–5 з пункту 1.2 є представниками класу $\mathcal{NP}$.

1.2. Довести, що якщо котрась із задач розпізнавання 3–5 з пункту 1.2 входить в клас $\mathcal{P}$, то відповідна задача обчислення розв'язується за поліноміальний час.

1.3. Довести, що коли з означення 1.1 вилучити обмеження $|u| \leq |w|^c$, то послаблене таким чином означення будуть задовольняти всі алгоритмічно розв'язні (за довільний час) мови.

1.4. Довести, що множина $\left\{ (x, n) : x \in \widetilde{\mathcal{Q}}_n \right\}$ належить класові $\mathcal{NP}$.

1.5. ([129]) Довести, що множина простих чисел належить класові $\mathcal{NP}$.

1.6. Довести, що разом з кожними двома мовами L_1 і L_2 клас $\mathcal{NP}$ містить також їх об'єднання та перетин.

1.7. Довести, що кожна мова з класу $\mathcal{NP}$ розпізнається за експоненційний час.

1.8. Припустимо, що мова $L \subseteq \{0,1\}^*$ розпізнається деяким ймовірнісним алгоритмом з однобічною помилкою (див. пункт III.3.1). Довести, що доповнення $\{0,1\}^* \setminus L$ належить класові $\mathcal{NP}$.

1.9. Нехай L_1 і L_2 — дві задачі з класу $\mathcal{NP}$, причому L_1 поліноміально зводиться до L_2. Припустимо, що задача L_1 є $\mathcal{NP}$-повною. Довести, що тоді задача L_2 також $\mathcal{NP}$-повна.

1.10. Довести, що будь-які дві $\mathcal{NP}$-повні задачі між собою поліноміально еквівалентні.

ЛІТЕРАТУРА

Посібниками з теорії $\mathcal{NP}$-повноти є [18] та [7] (див. також огляд [54]). Теорема Кука-Левіна доведена незалежно в [34] і [36]. В першій із цих робіт встановлено $\mathcal{NP}$-повноту задачі 1 з пункту 1.5, а $\mathcal{NP}$-повноту задач 2 і 3 доведено в [27] і [41] відповідно. В [18] міститься список понад 300 $\mathcal{NP}$-повних задач.

§ 2. Доведення без розголошення для мов у класі $\mathcal{NP}$

В § VII.6 ми ознайомились з двома конкретними протоколами доведення без розголошення — для квадратичності та неквадратичності. В загальному випадку доводяться твердження типу $z \in L$, де z — слово в деякому алфавіті, а L — мова в тому ж алфавіті. Якщо протокол доведення належності слова мові L володіє властивостями повноти, обґрунтованості і відсутності розголошення, то кажуть, що для L існує доведення без розголошення. Таким чином, множина квадратичних лишків $\{(x, n) : x \in \mathcal{Q}_n\}$, яку можна природним чином закодувати як мову у відповідному алфавіті, має досконале, а множина псевдоквадратів — статистичне доведення без розголошення. Природно виникає

питання, наскільки широким є клас мов, що володіють доведеннями без розголошення.

І множина квадратичних лишків, і множина псевдоквадратів є членами класу $\mathcal{NP}$ (див. пункт 1.2 і вправу 1.4). Цікаво зазначити, що саме членство мови L в класі $\mathcal{NP}$ можна інтерпретувати як можливість доведення твердження $w \in L$ для заданого слова w. Таке твердження справедливе тоді і тільки тоді, коли для нього існує доведення u, яке перевіряється за поліноміальний час (див. означення 1.1). Виявляється, що для кожної мови L з $\mathcal{NP}$ твердження типу $w \in L$ володіє доведенням без розголошення.

Із врахуванням зауваження 1.4 нескладно зрозуміти, що досить розробити доведення без розголошення для якоїсь конкретної $\mathcal{NP}$-повної мови. В якості такої ми виберемо множину всіх тих графів, які можна розмалювати трьома кольорами (вона відповідає задачі 2 з пункту 1.5).

Спочатку пояснимо ідею протоколу за допомогою його "фізичної" інтерпретації. Нехай задано граф $G(V, E)$ на множині вершин $V = \{1, \ldots, n\}$ з множиною ребер E. Боб хоче переконати Алісу, що вершини графа можна розмалювати трьома кольорами так, що будь-які дві суміжні вершини будуть різноколірними. При цьому Аліса не повинна отримати жодної інформації про саме розмалювання. (Нагадаємо, що самій Алісі не під силу визначити, чи граф володіє правильним розмалюванням, адже така задача розпізнавання є $\mathcal{NP}$-повною.) Процедуру доведення можна влаштувати таким чином.

- Боб знає деяке правильне розмалювання вершин графа G. Першим ділом, в цьому розмалюванні він міняє між собою всі три кольори випадковим чином. Наприклад, червоний стає голубим, голубий зеленим, а зелений червоним — всього 6 варіантів (деякі кольори можуть залишитись незміненими).
- На n пронумерованих картках Боб записує, яким кольором зафарбувалась відповідна вершина.
- Кожну з карток Боб замикає в окремій скриньці і всі n скриньок віддає Алісі, залишивши ключі від них в себе.
- Аліса вибирає випадкове ребро $\{u, v\} \in E$ і вимагає у Боба показати їй кольори, якими зафарбовані вершини u і v.
- Боб дає Алісі ключі від двох відповідних скриньок.
- Аліса відкриває їх і перевіряє, чи там записані два різні кольори. Якщо так, то Аліса залишається задоволеною, якщо ні, то звинувачує Боба в нечесності.

Вся процедура повторюється m^2 разів, де $m = \|E\|$ — кількість ребер графа G, причому кожен випадковий вибір здійснюється незалежно від попередніх. Якщо у кожному циклі результати перевірки були успішними, то Аліса переконується в істинності Бобового твердження.

Повнота. Зрозуміло, що якщо граф справді розмальовний трьома кольорами, а Боб знає правильне розмалювання і діє згідно з протоколом, то він переконує Алісу з імовірністю 1.

Обгрунтованість. Припустимо, що вершини графа неможливо розмалювати трьома кольорами, а Боб хитрує, записавши на картках неправильне розмалювання. Тоді принаймні для одного ребра $\{u, v\} \in E$ вершини u і v мають однакові кольори. Аліса попросить відкрити їй кольори саме цих вершин з імовірністю $1/m$. Отже, в одному циклі Боб зможе переконати Алісу з імовірністю щонайбільше $1 - 1/m$, а в кожному з m^2 циклів — з імовірністю щонайбільше $(1 - 1/m)^{m^2} \leq e^{-m}$, що є експоненційно малою величиною.

Відсутність розголошення. Якщо граф справді розмальовний трьома кольорами і Боб діє згідно з протоколом, то у кожному циклі Аліса отримує не що інше, як випадкову пару різних кольорів, яку вона і сама могла б змоделювати тривіальним чином. В цьому сенсі Аліса не довідується від Боба абсолютно нічого, хоча й переконується, що граф справді можна якось розмалювати.

Коли ми хочемо цю фізичну версію протоколу перетворити в "електронну", виникає питання, як можна обійтися без скриньок. З тією ж метою можна використати довільну важкооборотну перестановку $f : \{0,1\}^n \to \{0,1\}^n$ з ядром B (див. пункт VI.1.3). Ідея протоколу залишається тою ж.

Вхід: граф $G(V, E)$, де $V = \{1, \ldots, n\}$, $\|E\| = m$, і існує таке розфарбування $\phi : V \to \{1, 2, 3\}$, що $\phi(u) \neq \phi(v)$ для кожного ребра $(u, v) \in E$.

- Боб вибирає випадкову перестановку π множини кольорів $\{1, 2, 3\}$.
- Далі Боб формує розфарбування вершин графа G як послідовність кольорів $\pi(\phi(1)), \pi(\phi(2)), \ldots, \pi(\phi(n))$, яку він природним чином записує у вигляді двійкової послідовності $\sigma_1, \sigma_2, \ldots, \sigma_{2n-1}, \sigma_{2n}$. При цьому колір $\pi(\phi(v))$ вершини $v \in V$ кодується двома бітами σ_{2v-1} і σ_{2v}.
- Це розфарбування Боб шифрує, вибираючи випадковим чином $r_i \in \{0,1\}^n$ і покладаючи $s_i = f(r_i)$ та $b_i = B(r_i) \oplus \sigma_i$ для всіх $1 \leq i \leq 2n$. Послідовність $(s_1, b_1), \ldots, (s_{2n}, b_{2n})$ Боб посилає Алісі.

- Аліса вибирає випадкове ребро $\{u, v\} \in E$ і посилає його Бобові, вимагаючи показати їй кольори, якими зафарбовані вершини u і v.
- Боб посилає Алісі "ключі" $r_{2u-1}, r_{2u}, r_{2v-1}, r_{2v}$.
- Аліса перевіряє рівності $s_j = f(r_j)$ для $j \in \{2u-1, 2u, 2v-1, 2v\}$ (справжність ключів), і якщо вони виконуються, то визначає $\sigma_j = b_j \oplus B(r_j)$ і перевіряє умову $\pi(\phi(u)) \neq \pi(\phi(v))$ (правильність розмалювання). Якщо перевірка здійснюється успішно, то Аліса залишається задоволеною, якщо ж ні, то звинувачує Боба в нещирості.

Вся процедура повторюється m^2 разів, причому щоразу кожен випадковий вибір здійснюються незалежно від попередніх. Якщо в кожному циклі результати перевірки успішні, то Бобові щастить переконати Алісу в існуванні правильного розмалювання вершин графа G.

Повнота та обгрунтованість цього протоколу встановлюється дослівно тими ж міркуваннями, що й для його фізичної версії. Розглянемо властивість відсутності розголошення. На інтуїтивному рівні її можна аргументувати так. Позаяк предикат B є ядром функції f, за s_i Аліса не може визначити біт $B(r_i)$, а відтак за s_i і b_i вона не здатна визначити σ_i, тобто з отриманої під час протоколу інформації Аліса неспроможна *ефективним чином* визначити розмалювання.

Не претендуючи на повне доведення, сформулюємо властивість відсутності розголошення для розглянутого протоколу формально. Нехай Аліса вибирає своє повідомлення $\{u, v\}$ до Боба за допомогою деякого поліноміального ймовірнісного алгоритму A на основі отриманої послідовності $(s_1, b_1), \ldots, (s_{2n}, b_{2n})$. Позначимо через $VIEW_G = \langle (s_1, b_1), \ldots, (s_{2n}, b_{2n}), \{u, v\}, r_{2u-1}, r_{2u}, r_{2v-1}, r_{2v} \rangle$ випадкову величину, яку Аліса спостерігає упродовж протоколу. Тоді існує поліноміальний ймовірнісний алгоритм M_A такий, що якщо граф G справді розмальовний трьома кольорами, випадкові величини $M_A(G)$ і $VIEW_G$ нерозрізнювані за поліноміальний час. Тобто для будь-якого поліноміального ймовірнісного алгоритму D, який подає на вихід одне із двох значень 1 або 0,

$$|\mathbf{P}[D(VIEW_G) = 1] - \mathbf{P}[D(M_A(G)) = 1]| < \frac{1}{n^c}$$

для довільної константи c при досить великих значеннях n, де n — кількість вершин графа G.

Протокол з такою властивістю називають доведенням без розголошення *в обчислювальному сенсі*.

ВПРАВИ

2.1. Довести, що кожне досконале доведення без розголошення є статистичним, а кожне статистичне у свою чергу є доведенням без розголошення в обчислювальному сенсі.

ЛІТЕРАТУРА

Доведення без розголошення для розмальовності графа в 3 кольори запропоноване в [98]. Подальший розвиток досліджень в цій галузі можна прослідкувати за монографією [93] та роботами [74, 87].

Додаток А.

Таблички частот

␣	0.134	е	0.043	л	0.028	ч	0.011	ї	0.006
о	0.082	р	0.038	у	0.027	б	0.010	є	0.006
н	0.070	і	0.037	п	0.025	х	0.010	ф	0.005
а	0.070	с	0.036	я	0.021	ц	0.009	ш	0.005
и	0.056	к	0.036	з	0.019	ю	0.009	щ	0.003
т	0.051	м	0.033	ь	0.015	ж	0.008	ґ	0.000
в	0.046	д	0.028	г	0.013	й	0.007		

Таблиця А.1. Частоти літер та пропуску між словами в українській мові.

Таблиця А.1 отримана І. Я. Берегуляком [8] шляхом статистичного аналізу тексту з комп'ютерної лінгвістики розміром 1 Mb. Цікаво співставити її з аналогічними таблицями для української мови в інших джерелах [5, 46].

Таблиця А.2 запозичена з [23]. Таблиця А.3 отримана в результаті статистичного аналізу статті про криптографію в Британській енциклопедії [141] і наведена там же. Цікаво порівняти її з іншими джерелами, наприклад з таблицею частот для англійської мови з [143], яка відтворена в [3].

Таблиці А.4 та А.5 отримані в [8] з використанням того ж тексту, що й таблиця А.1.

␣	0.175	р	0.040	я	0.018	х	0.009
о	0.090	в	0.038	э	0.016	ж	0.007
е,ё	0.072	л	0.035	і	0.016	ш	0.006
а	0.062	к	0.028	б	0.014	ю	0.006
и	0.062	м	0.026	ь,ъ	0.014	ц	0.004
н	0.053	д	0.025	г	0.013	щ	0.003
т	0.053	п	0.023	ч	0.012	э	0.003
с	0.045	у	0.021	й	0.010	ф	0.002

Таблиця А.2. Частоти літер та пропуску між словами в російській мові.

e	0.127	n	0.067	l	0.040	m	0.024	v	0.008
t	0.097	s	0.067	d	0.031	f	0.021	k	0.008
i	0.075	r	0.064	p	0.030	b	0.017	x	0.005
a	0.073	h	0.049	y	0.027	g	0.016	q	0.002
o	0.068	c	0.045	u	0.024	w	0.013	z	0.001
								j	0.001

Таблиця А.3. Частоти літер в англійській мові.

␣п	0.016	ст	0.013	а␣	0.011	␣н	0.011
␣в	0.015	о␣	0.013	я␣	0.011	у␣	0.010
и␣	0.014	на	0.012	но	0.011	ан	0.010

Таблиця А.4. Частоти найпоширеніших в українській мові біграм із врахуванням пропуску між словами (розділові знаки ігноруються).

ст	0.017	ан	0.014	ко	0.012	нн	0.011
на	0.016	ов	0.013	ви	0.011	ен	0.010
но	0.014	ти	0.013	ни	0.011	ма	0.010

Таблиця А.5. Частоти найпоширеніших в українській мові біграм у словах.

Додаток Б.

Математичний апарат

Б.1. Множини. В підручнику активно використовується стандартна теоретико-множинна термінологія. Цілком достатнім є "наївне" уявлення про *множину* як про деяку сукупність об'єктів — *елементів* множини. Множина X, що складається з n елементів $x_1, \ldots, x_n$, може задаватись таким записом: $X = \{x_1, \ldots, x_n\}$. Якщо x є елементом множини X (або множина X містить x), то пишемо $x \in X$.

Множина Y називається *підмножиною* множини X, якщо X містить всі елементи множини Y. Для множини X, що складається з n елементів, кількість її k-елементних підмножин позначається символом $\binom{n}{k}$ (кількість сполук із n по k). Справедлива формула $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

Для множин X та Y, через $X \cup Y$ позначається їх *об'єднання* — множина, що містить елементи обох множин і лише їх. *Перетин* $X \cap Y$ цих двох множин складається з їх спільних елементів. Різниця $X \setminus Y$ складається з тих елементів множини X, яких немає в Y.

Декартів добуток $X \times Y$ складається зі всіх впорядкованих пар (x, y), де $x \in X$ і $y \in Y$. Якщо X та Y налічують n та m елементів відповідно, то в $X \times Y$ міститься nm елементів. *Декартів степінь* X^m складається зі всеможливих послідовностей $(x_1, \ldots, x_m)$, де $x_i \in X$. Кількість таких послідовностей дорівнює n^m, де n — кількість елементів у множині X. Зокрема, кількість двійкових послідовностей, тобто елементів множини $\{0,1\}^m$, дорівнює 2^m.

Б.2. Відображення. Нехай X і Y — дві множини. Припустимо, що кожному елементові x множини X поставлено у відповідність деякий елемент $y = f(x)$ множини Y. В такому випадку кажемо, що задано *відображення* або *функцію* $f : X \to Y$ із множини X у множину Y. Відображення $f : X \to Y$ називається *ін'єктивним*, якщо воно різним аргументам співставляє різні значення: $f(x_1) \neq f(x_2)$ для $x_1 \neq x_2$. Відображення $f : X \to Y$ *сюр'єктивне*, якщо кожен елемент y з множини Y має прообраз — такий елемент

$x \in X$, що $f(x) = y$. *Бієктивним* є відображення, яке ін'єктивне і сюр'єктивне одночасно. Для двох відображень $f : X \to Y$ та $g : Y \to Z$ їх *композиція* $g \circ f : X \to Z$ задається співвідношенням $g \circ f(x) = g(f(x))$ для $x \in X$. *Тотожне* відображення $\mathrm{id}_X : X \to X$ залишає елементи множини X на місці: $\mathrm{id}_X(x) = x$. Відображення $g : Y \to X$ вважається *лівим оберненим* до відображення $f : X \to Y$ за умови, що їх композиція $g \circ f$ рівна id_X; і *правим оберненим* за умови, що $f \circ g = \mathrm{id}_Y$. Відображення g називається *оберненим* до f, якщо воно є одночасно і лівим, і правим оберненим до f.

ТЕОРЕМА ПРО ОБЕРНЕНЕ ВІДОБРАЖЕННЯ.

1) *Відображення $f : X \to Y$ ін'єктивне тоді і лише тоді, коли до нього існує ліве обернене.*
2) *Відображення $f : X \to Y$ сюр'єктивне тоді і лише тоді, коли до нього існує праве обернене.*
3) *Якщо відображення $f : X \to Y$ бієктивне, то його ліве обернене збігається із правим оберненим.*
4) *У випадку, коли множини X та Y скінченні і містять однакову кількість елементів, відображення $f : X \to Y$ має ліве обернене тоді і тільки тоді, коли воно має праве обернене.* ∎

Б.3. Групи. *Групою* називається множина G, наділена бінарною операцією $\star$ з такими властивостями:

1) $(x \star y) \star z = x \star (y \star z)$ для будь-яких елементів $x, y, z \in G$ *(асоціативність)*;
2) в G існує *нейтральний* елемент e такий, що $x \star e = e \star x = x$ для всіх $x \in G$;
3) для кожного елемента $x \in G$ в G є *обернений* елемент x' такий, що $x \star x' = x' \star x = e$.

Якщо підмножина H множини G утворює групу відносно тієї ж операції $\star$, то вона називається *підгрупою* групи G. Так, множина раціональних чисел $\mathbb{Q}$ утворює групу за додаванням ($e = 0$, $x' = -x$), а множина цілих чисел $\mathbb{Z}$ є її підгрупою. Множина додатніх раціональних чисел $\mathbb{Q}_+$ утворює групу за множенням ($e = 1$, $x' = 1/x$). Якщо групову операцію називаємо множенням, то саму групу називаємо *мультиплікативною*, а її нейтральний елемент — одиницею. Якщо ж операцію називаємо додаванням, то групу називаємо *адитивною*, нейтральний елемент нулем, а обернений елемент — протилежним елементом.

Для елемента x групи G через x^i позначається його i-тий степінь — елемент $x \star x \star \ldots \star x$, де операція виконана $i - 1$ раз. (В адитивній формі те ж саме записується як $ix = x + \ldots + x$.) Нескладною вправою є доведення того, що для кожного елемента x скінченної групи для деякого показника m виконується рівність $x^m = e$. Найменше з таких m називається *порядком елемента* x у групі G.

Порядком скінченної групи називається кількість її елементів. Легко зауважити, що всі степені елемента групи утворюють у ній підгрупу, порядок якої дорівнює порядку елемента.

Теорема Лагранжа (1771).

1) *Порядок підгрупи є дільником порядку групи.*

2) *Порядок елемента є дільником порядку групи.* ▪

Нехай маємо дві групи G і G' з операціями $\star$ і $\circ$ відповідно. Кажемо, що відображення $f : G \to G'$ *зберігає операцію*, якщо $f(x \star y) = f(x) \circ f(y)$ для всіх $x, y \in G$. Таке відображення $f : G \to G'$ називається *гомоморфізмом* з групи G у групу G'. *Ядром* гомоморфізму $f : G \to G'$ є множина всіх тих елементів групи G, які f відображає в нейтральний елемент групи G'. Наприклад, відображення, яке кожному цілому числу ставить у відповідність його остачу від ділення на натуральне n, є гомоморфізмом із адитивної групи $\mathbb{Z}$ в адитивну групу $\mathbb{Z}_n$, ядро якого утворюють цілі числа, кратні n. Легко показати, що ядро гомоморфізму $f : G \to G'$ утворює в G підгрупу.

Гомоморфізм $f : G \to G'$ є ін'єктивним тоді і лише тоді, коли його ядро складається тільки із нейтрального елемента групи G. Таке ядро називається *тривіальним*.

Гомоморфізм, який є бієктивним відображенням, називається *ізоморфізмом*. Дві групи *ізоморфні*, якщо існує ізоморфізм з однієї з них на іншу. Наприклад, двійковий логарифм $\log_2 : \mathbb{R}_+ \to \mathbb{R}$ задає ізоморфізм із мультиплікативної групи невід'ємних дійсних чисел $\mathbb{R}_+$ в адитивну групу всіх дійсних чисел $\mathbb{R}$. Ізоморфні групи мають тотожні алгебраїчні властивості.

Для груп G_1 і G_2 з операціями $\star$ і $\circ$ відповідно, через $G_1 \times G_2$ позначаємо їх *прямий добуток* — множину пар (x_1, x_2), де $x_1 \in G_1$ а $x_2 \in G_2$, із *покомпонентним* виконанням операцій. А саме, результатом виконання операції над елементами (x_1, x_2) і (y_1, y_2) множини $G_1 \times G_2$ вважається елемент $(x_1 \star y_1, x_2 \circ y_2)$. Нескладно перевірити, що відносно введеної операції прямий добуток груп є групою.

Група називається *комутативною* або *абелевою*, якщо групова операція володіє властивістю *комутативності*: $x \star y = y \star x$ для будь-яких елементів x та y.

Якщо кожен елемент групи G є степенем її елемента g, то цей елемент називають *твірним* і кажуть, що він *породжує* групу G. Якщо група G має порядок n, то g є її твірним елементом тоді і лише тоді, коли його порядок теж дорівнює n. Група, яка має твірний елемент, називається *циклічною*. Циклічна група порядку n ізоморфна групі $\mathbb{Z}_n$, а нескінченна циклічна група — групі $\mathbb{Z}$.

Твердження Б.1. *Циклічна група порядку n має рівно $\phi(n)$ твірних елементів.*

Доведення. Його досить провести для $\mathbb{Z}_n$. Покажемо, що g породжує $\mathbb{Z}_n$ тоді і лише тоді, коли числа g і n взаємно прості. Справді, якщо НСД$(g, n) = 1$, то за твердженням II.2.2 маємо $ug + vn = 1$ для деяких цілих u і v. Із цієї рівності випливає, що кожен елемент $a \in \mathbb{Z}_n$ дорівнює au-кратній сумі $g + \ldots + g$ (можна вважати, що $u > 0$, а інакше коефіцієнт u слід поміняти на його остачу від ділення на n).

Навпаки, якщо НСД$(g, n) = m$ і $m > 1$, то $(n/m)g = 0$ в $\mathbb{Z}_n$. Отже, порядок g не перевищує $n/m < n$, і g не може бути твірним. ▪

Б.4. Група перестановок. Іншим прикладом групи є множина $\mathrm{Sym}\, X$ бієктивних відображень множини X на себе з операцією композиції. Нейтральним елементом є тотожне відображення id_X, а оберненим елементом до $f \in \mathrm{Sym}\, X$ є відображення f', обернене до f. Бієкцію множини на себе називають ще *перестановкою* цієї множини. Відповідно, $\mathrm{Sym}\, X$ називається *групою перестановок* множини X. Якщо множина X налічує n елементів, то очевидним чином група $\mathrm{Sym}\, X$ ізоморфна групі $\mathrm{Sym}\{1, 2, \ldots, n\}$. Остання називається *симетричною групою степеня n* і позначається через S_n.

Перестановку σ із S_n звично записують у вигляді таблички

$$\begin{pmatrix} 1 & 2 & \ldots & n \\ \sigma(1) & \sigma(2) & \ldots & \sigma(n) \end{pmatrix}.$$

Цикл $(i_1 i_2 \ldots i_l)$, де $i_1, i_2, \ldots, i_l$ різні числа від 1 до n, — це перестановка, яка елемент i_j, $j < l$, відображає в i_{j+1}, i_l в i_1, а всі інші елементи самі в себе. l називається *довжиною* циклу. Цикли $(i_1 i_2 \ldots i_l)$ та $(i'_1 i'_2 \ldots i'_{l'})$ *незалежні*, якщо множини елементів $\{i_1, i_2, \ldots, i_l\}$ та $\{i'_1, i'_2, \ldots, i'_{l'}\}$ не перетинаються. Кожна перестановка є композицією (або добутком) попарно незалежних циклів. Наприклад, перестановка

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 2 & 6 & 3 & 1 & 4 \end{pmatrix}$$

дорівнює добутку (15)(364). Якщо перестановка σ розкладається в добуток m незалежних циклів з довжинами $l_1, l_2, \ldots, l_m$, то її *знак* $\varepsilon(\sigma)$ обчислюється за формулою

$$\varepsilon(\sigma) = (-1)^{\sum_{j=1}^{m} (l_j - 1)}.$$

Знак тотожної перестановки приймається рівним 1.

Група S_n має порядок $n!$. Порядок циклу довжини l як елемента групи S_n дорівнює l. Порядок перестановки, яка є добутком m незалежних циклів з довжинами $l_1, l_2, \ldots, l_m$, дорівнює НСК цих чисел.

Б.5. Кільця. *Кільцем* називається множина R з двома заданими на ній операціями, $+$ (додавання) та $\cdot$ (множення), яка має такі властивості:

1) відносно додавання R утворює абелеву групу;

2) операція множення асоціативна;

3) множення дистрибутивне за додаванням, що означає виконання рівностей

$$(x+y)\cdot z = x\cdot z + y\cdot z, \quad z\cdot(x+y) = z\cdot x + z\cdot y$$

для всіх $x, y, z \in R$.

Якщо окрім того операція множення комутативна, то кільце називається *комутативним*. R називається *кільцем з одиницею*, якщо в ньому є нейтральний відносно множення елемент. Через 1 позначаємо нейтральний елемент для множення, а через 0 — для додавання. Прикладом комутативного кільця з одиницею є множина $\mathbb{Z}$ цілих чисел зі звичайним додаванням та множенням. Прикладом некомутативного кільця з одиницею є кільце матриць, яке розглядається нижче.

Елемент $x \in R$ кільця з одиницею називається *оборотнім справа [зліва]*, якщо $x \cdot x' = 1$ $[x' \cdot x = 1]$ для деякого $x' \in R$. Кажуть, що елемент x' є *правим [лівим] оберненим* до x. Елемент x називається *оборотнім* або *дільником одиниці*, якщо він оборотній і зліва, і справа. Кожен оборотній елемент має єдиний лівий і єдиний правий обернені елементи, які рівні між собою. Цей єдиний елемент називається оберненим до x і позначається через x^{-1}. Для оборотних елементів x і y неважко перевірити рівність $(x\cdot y)^{-1} = y^{-1}\cdot x^{-1}$. Звідси випливає, що множина всіх оборотних елементів кільця з одиницею R утворює групу, яка називається *мультиплікативною групою кільця* R і позначається через R^*. Наприклад, $\mathbb{Z}^* = \{1, -1\}$. Зрозуміло, що мультиплікативна група комутативного кільця сама є комутативною.

Відображення $f : R \to R'$ називається *гомоморфізмом*, якщо воно зберігає операції додавання та множення. Для кілець з одиницею повинна виконуватись ще одна умова: f має відображати одиницю кільця R в одиницю кільця R'. *Ядром* гомоморфізму $f : R \to R'$ є множина всіх тих елементів кільця R, які f відображає в нуль кільця R'. Прикладом гомоморфізму кілець є відображення із $\mathbb{Z}$ в $\mathbb{Z}_n$, яке кожному цілому числу ставить у відповідність його остачу від ділення на натуральне n. Ядро утворюють цілі числа, кратні n. Як і у випадку груп, гомоморфізм $f : R \to R'$ є ін'єктивним тоді і лише тоді, коли його ядро *тривіальне*, тобто складається тільки із нуля кільця R. Гомоморфізм, який є бієктивним відображенням, називається *ізоморфізмом*. Кільця R і R' *ізоморфні*, якщо існує ізоморфізм з R на R'.

Для кілець R_1 і R_2 через $R_1 \oplus R_2$ позначаємо їх *прямий добуток* — множину пар (x_1, x_2), де $x_1 \in R_1$ а $x_2 \in R_2$, із покомпонентним додаванням та множенням. Неважко впевнитись, що прямий добуток кілець є кільцем. Простим наслідком означень є

Твердження Б.2. *Мультиплікативні групи $(R_1 \oplus R_2)^*$ і $R_1^* \times R_2^*$ ізоморфні.*

Б.6. Кільце матриць. У цьому пункті R позначає кільце з одиницею. *Матрицею розміру $k \times m$* над R називається індексована сукупність (a_{ij}) елементів з R, де $1 \le i \le k$, $1 \le j \le m$. Елементи (a_{ij}) називають *коефіцієнтами* матриці і звичайно розміщують у вигляді

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \dots & a_{km} \end{pmatrix},$$

так що матриця розміру $k \times m$ складається із k *рядків* та m *стовпчиків*. Матриця розміру $k \times 1$ називається *вектором-стовпчиком*, а розміру $1 \times k$ — *вектором-рядком*.

Сумою матриць $A = (a_{ij})$ та $B = (b_{ij})$ однакового розміру є матриця $C = (c_{ij})$ того ж розміру з коефіцієнтами $c_{ij} = a_{ij} + b_{ij}$. Пишемо $C = A + B$. *Добутком* матриці $A = (a_{is})$ розміру $k \times l$ на матрицю $B = (b_{sj})$ розміру $l \times m$ є матриця $C = (c_{ij})$ розміру $k \times m$ з коефіцієнтами $c_{ij} = \sum_{s=1}^{l} a_{is}b_{sj}$. Пишемо $C = AB$. Нескладно перевірити, що операція множення матриць є асоціативною. Результатом *множення матриці $A = (a_{ij})$ на елемент $d \in R$* є матриця dA з коефіцієнтами (da_{ij}).

Матриця розміру $k \times k$ називається *квадратною* матрицею *порядку* k. Коефіцієнти a_{ii}, $i \le k$, квадратної матриці (a_{ij}) називаються *діагональними*. Квадратні матриці заданого порядку k відносно операцій додавання та множення утворюють кільце, яке ми позначаємо $\mathrm{M}_k(R)$. Це кільце з одиницею, в ролі якої виступає *одинична* матриця I_k, діагональні коефіцієнти якої рівні одиниці кільця R, а всі інші — нулю.

Твердження Б.3. *Кільця $\mathrm{M}_k(\mathrm{M}_l(R))$ і $\mathrm{M}_{kl}(R)$ ізоморфні.*

Це означає, що матриці порядку kl можна розбити на блоки розміру l на l, після чого додавати і множити такі матриці поблоково.

$\mathrm{M}_k(R)^*$, мультиплікативна група оборотних матриць, має назву *повної лінійної групи* і позначається також як $\mathrm{GL}_k(R)$.

Далі кільце R вважається комутативним.

Визначник $\det A$ квадратної матриці $A = (a_{ij})$ порядку k дорівнює наступному виразу:

$$\det A = \sum_{\sigma} \varepsilon(\sigma) a_{1\sigma(1)} a_{2\sigma(2)} \dots a_{k\sigma(k)},$$

де сумування проводиться за всіма перестановками σ з S_n, а $\varepsilon(\sigma)$ означає знак перестановки.

Твердження Б.4. *Визначник добутку матриць дорівнює добутку їх визначників:* $\det AB = \det A \det B$. ∎

Якщо взяти в останній рівності в якості B матрицю A', праву обернену до A, то отримаємо $\det A \det A' = \det I_k = 1$. Подібна рівність справедлива і для оберненої зліва матриці. Отже,

якщо матриця A оборотна зліва або справа,

то $\det A \in R^*$. (1)

Алгебраїчним доповненням елемента a_{ij} матриці A називається значення $A_{ij} = (-1)^{i+j} M_{ij}$, де M_{ij} — визначник матриці, яка отримується із матриці A після викреслення її i-го рядка та j-го стовпчика. Має місце

ФОРМУЛА ОБЕРНЕНОЇ МАТРИЦІ. *Для матриці $A' = (a'_{ij})$ з коефіцієнтами $a'_{ij} = A_{ji}$ виконуються співвідношення $AA' = A'A = (\det A) I_k$. Отже, в матриці A^{-1}, оберненій до A, коефіцієнт з індексами ij дорівнює $A_{ji}(\det A)^{-1}$* (порядок індексів помінявся!).

З формули випливає імплікація, що доповнює (1): якщо $\det A \in R^*$, то матриця A має обернену. Зафіксуємо отриманий зв'язок у наступному твердженні.

ТВЕРДЖЕННЯ Б.5.

1) *$A \in \mathrm{M}_k(R)^*$ тоді і лише тоді, коли* $\det A \in R^*$.

2) *Якщо матриця $A \in \mathrm{M}_k(R)$ має праву або ліву обернену, то вона оборотня, тобто має матрицю, що є одночасно і правою, і лівою оберненою.*

За твердженням Б.4 відображення $\det : \mathrm{M}_k(R)^* \to R^*$, яке співставляє матриці її визначник, є гомоморфізмом мультиплікативних груп. Ядром цього гомоморфізму є підгрупа матриць з визначником 1, яка має назву *спеціальної лінійної групи* і позначається $\mathrm{SL}_k(R)$.

Б.7. Поля. *Полем* називається множина F з двома заданими на ній операціями, $+$ (додавання) та $\cdot$ (множення), яка має такі властивості:

1) відносно додавання F утворює абелеву групу з нейтральним елементом 0;

2) відносно множення $F^* = F \setminus \{0\}$ утворює абелеву групу з нейтральним елементом 1;

3) множення дистрибутивне за додаванням.

ТВЕРДЖЕННЯ Б.6. *Мультиплікативна група F^* скінченного поля є циклічною.* ∎

Для повноти наведемо і такий факт.

ТЕОРЕМА ГАУСА. *Група $\mathbb{Z}_n^*$ є циклічною для значень $n = 1, 2, 4, p^k, 2p^k$, де число p непарне просте, а k натуральне, і лише для них.*

Б.8. Кільце многочленів. *Многочлен (або поліном) степеня n від однієї змінної x* над комутативним кільцем з одиницею R звичайно зображується у вигляді арифметичного виразу $a_n x^n + \ldots + a_2 x^2 + a_1 x + a_0$, де $a_n, \ldots, a_2, a_1, a_0 \in R$ — *коефіцієнти* многочлена, причому *старший* коефіцієнт a_n відмінний від 0. Відносно стандартних операцій додавання та множення многочлени над R утворюють кільце, яке позначається $R[x]$. Далі у цьому пункті розглядається кільце $F[x]$ многочленів над деяким полем F.

У кільці $F[x]$ стандартним чином вводиться операція ділення з остачею. Завдяки цьому у кільці $F[x]$ цілком аналогічно до кільця $\mathbb{Z}$ працює алгоритм Евкліда, який дозволяє знаходити найбільший спільний дільник двох многочленів. Подібно до того, як це було зроблено у пункті II.2.2, доводиться теорема про однозначність розкладу на множники в $F[x]$. Роль простих чисел відіграють *незвідні многочлени*, тобто многочлени ненульового степеня, які не мають дільників меншого (але ненульового) степеня. Теорема говорить, що кожен многочлен ненульового степеня над полем F розкладається у добуток незвідних многочленів однозначно, якщо не брати до уваги порядок співмножників і не розрізняти співмножники, які можуть бути отриманими один із одного домноженням на елемент поля.

Нехай $p(x)$ — деякий многочлен над F. Для $b \in F$ через $p(b)$ позначимо елемент поля F, який отримується як результат підстановки елемента b у многочлен замість змінної x і виконання операцій множення і додавання в F. Елемент $p(b)$ будемо називати *значенням многочлена $p(x)$ в точці b*. Якщо $p(b) = 0$, то b називається *коренем* многочлена $p(x)$. Діленням з остачею многочлена $p(x)$ на многочлен $x - b$ і підстановкою $x = b$ в отриману рівність доводиться

ТЕОРЕМА БЕЗУ. *Якщо b — корінь ненульового многочлена $p(x)$ над полем F, то $p(x)$ ділиться на многочлен $x - b$.* ∎

Простим наслідком цієї теореми є

ТВЕРДЖЕННЯ Б.7. *Ненульовий многочлен $p(x)$ степеня n над полем F має не більше, ніж n коренів.* ∎

Із твердження випливає, що два многочлени степеня не вищого від n кожен, значення яких збігаються в $n+1$ різних точках, мусять бути рівними, оскільки їх різниця є многочленом степеня не більшого від n, який має $n+1$ корінь. Отже, щонайбільше один многочлен степеня не більшого ніж n може набувати в $n+1$ заданій точці $b_0, \ldots, b_n$ задані значення $p(b_0), \ldots, p(b_n)$. Такий многочлен справді існує і дається наступною формулою.

ІНТЕРПОЛЯЦІЙНА ФОРМУЛА ЛАГРАНЖА.

$$p(x) = \sum_{i=0}^{n} p(b_i) \prod_{j:j\neq i} \frac{x - b_j}{b_i - b_j}.$$

Б.9. Векторні простори. *Лінійним* (або *векторним*) *простором над полем* F називається абелева група V із груповою операцією $+$ та операцією множення елемента групи (*вектора*) на елемент поля (*скаляр*), результатом виконання якої є вектор. Операції мають задовольняти такі умови.

1) множення унітарне: $1v = v$, де 1 — одиниця поля F, а v — довільний вектор з V;
2) множення асоціативне: $a(bv) = (ab)v$ для всіх $a, b \in F$ і $v \in V$;
3) множення і додавання пов'язані законами дистрибутивності:

$$a(v_1 + v_2) = av_1 + av_2, \quad (a_1 + a_2)v = a_1v + a_2v$$

для всіх $a, a_1, a_2 \in F$ і $v, v_1, v_2 \in V$.

Наприклад, множина F^k для довільного натурального k із покомпонентним додаванням та покомпонентним множенням на елемент поля F є лінійним простором над F.

Для векторів $v_1, \ldots, v_k \in V$ вектор $\sum_{i=1}^{k} a_i v_i$ називається їхньою *лінійною комбінацією* з коефіцієнтами $a_1, \ldots, a_k \in F$. Всеможливі лінійні комбінації векторів $v_1, \ldots, v_k$ утворюють їхню *лінійну оболонку*. Вектори $v_1, \ldots, v_k$ є *повною системою*, якщо їх лінійна оболонка охоплює весь простір V. Вектори $v_1, \ldots, v_k$ називаються *лінійно незалежними*, якщо жоден з них не належить лінійній оболонці решти. *Базою* простору V називається така система векторів $v_1, \ldots, v_k$, що кожен вектор з V можна подати як лінійну комбінацію векторів цієї системи, причому однозначним чином. Зрозуміло, що база повинна бути лінійно незалежною і повною системою. Окрім того, справедливе

ТВЕРДЖЕННЯ Б.8. *Для системи векторів $S = \{v_1, \ldots, v_k\}$ наступні три умови еквівалентні:*

1) *S є базою;*
2) *S є мінімальною повною системою (іншими словами, якщо вилучити із S будь-який вектор, система перестане бути повною);*
3) *S є максимальною лінійно незалежною системою (іншими словами, якщо добавити до S будь-який вектор, система перестане бути лінійно незалежною).* ∎

Із твердження видно, що кожен скінченний лінійний простір V має деяку базу $v_1, \ldots, v_k$. Припустимо, що поле F також скінченне. З означення бази випливає, що V складається із q^k векторів, де q — кількість елементів поля F. Отже, будь-які дві бази мусять мати однакову кількість векторів (цей факт має місце і для нескінченних полів, якщо лише лінійний простір має скінченну базу). Кількість векторів у базі називається *вимірністю* лінійного простору.

Будь-яка підмножина простору V, замкнена відносно операцій, сама є лінійним простором над тим же полем і називається *лінійним підпростором* простору V. Очевидно, лінійна оболонка системи із l лінійно незалежних векторів є l-вимірним підпростором.

Матриця з $\mathrm{M}_k(F)$ називається *невиродженою*, якщо її стовпчики є лінійно незалежними векторами лінійного простору F^k.

ТВЕРДЖЕННЯ Б.9. *Квадратна матриця над полем невироджена тоді і тільки тоді, коли вона оборотна.*

ЛІТЕРАТУРА

За докладним викладом математичних основ читач скеровується до підручників [11, 37, 29, 30, 31, 6]. Кращий спосіб обернення матриці над полем, ніж пряме використання формули оберненої матриці, пояснюється, наприклад, в [30, пункт 7 в § 3 розділу 2]. Доведення теореми Гауса можна знайти в [13, пункт d в § 7] або в [2, твердження 4.1.3].

Додаток В.

Колекція нерівностей

НЕРІВНІСТЬ В.1. (ВАРІАНТ ФОРМУЛИ СТІРЛІНГА)

$$n^n e^{-n}\sqrt{2\pi n}e^{1/(12n+1)} < n! < n^n e^{-n}\sqrt{2\pi n}e^{1/(12n)}.$$

ДОВЕДЕННЯ. Див. [58, співвідношення (9:14) у розділі II]. ∎

Дійснозначна функція $f : (a,b) \to \mathbb{R}$ називається *опуклою* на інтервалі (a,b), якщо для довільних $x_1, x_2 \in (a,b)$ і $\alpha_1, \alpha_2 \geq 0$ таких, що $\alpha_1 + \alpha_2 = 1$, виконується нерівність $f(\alpha_1 x_1 + \alpha_2 x_2) \leq \alpha_1 f(x_1) + \alpha_2 f(x_2)$. Наочно це означає, що хорда, яка сполучає дві точки на графіку, знаходиться над ним. Якщо виконується обернена нерівність, тобто хорда знаходиться під графіком, то функція називається *увігнутою* або *опуклою вверх*.

НЕРІВНІСТЬ В.2. (ЄНСЕНА) *Якщо $f : (a,b) \to \mathbb{R}$ — опукла функція, то для будь-яких $x_1, \ldots, x_n \in (a,b)$ і невід'ємних $\alpha_1, \ldots, \alpha_n$ з $\alpha_1 + \ldots + \alpha_n = 1$ справедлива нерівність*

$$f(\alpha_1 x_1 + \ldots + \alpha_n x_n) \leq \alpha_1 f(x_1) + \ldots + \alpha_n f(x_n).$$

Відповідно у випадку увігнутої функції виконується обернена нерівність.

ДОВЕДЕННЯ. Індукцією за n. ∎

НЕРІВНІСТЬ В.3. (МІЖ СЕРЕДНІМ ГЕОМЕТРИЧНИМ І СЕРЕДНІМ АРИФМЕТРИЧНИМ) *Для невід'ємних $x_1, \ldots, x_n$ має місце нерівність*

$$\sqrt[n]{x_1 \cdot \ldots \cdot x_n} \leq \frac{x_1 + \ldots + x_n}{n}.$$

ДОВЕДЕННЯ. Випливає з нерівності Єнсена для увігнутої функції $f(x) = \ln x$ при $\alpha_1 = \ldots = \alpha_n = \frac{1}{n}$. ∎

Наступна нерівність є простою, але практичною.

НЕРІВНІСТЬ В.4. *Для довільного дійсного x,*

$$1 - x \leq e^{-x}.$$

ДОВЕДЕННЯ. Для доведення нерівність зручніше переписати у вигляді $1+x \leq e^x$. Легко переєвідчитись, що функція $f(x) = e^x - x - 1$ має абсолютний мінімум у точці $x = 0$, де $f(x) = 0$. Або ж можна показати, що пряма $y = 1+x$ є дотичною до графіка опуклої функції $y = e^x$ в точці $x = 0, y = 1$. ∎

Нам згодяться ще дві прості нерівності.

НЕРІВНІСТЬ В.5. *Для $x \geq 0$,*

$$\ln(1+x) \geq x - x^2/2.$$

ДОВЕДЕННЯ. Розглянемо функцію $f(x) = \ln(1+x) - x + x^2/2$. Продиференціювавши, знаходимо $f'(x) = \frac{x^2}{1+x}$. Як бачимо, функція $f(x)$ при $x \geq 0$ є зростаючою, а отже найменшим її значенням у цій множині є $f(0) = 0$. ∎

НЕРІВНІСТЬ В.6. *Для $0 \leq x < 1$,*

$$\ln(1-x) \geq -x - \frac{x^2}{2(1-x)}.$$

ДОВЕДЕННЯ. Розглянемо функцію $f(x) = \ln(1-x) + x + \frac{x^2}{2(1-x)}$. Знаходимо $f'(x) = \frac{x^2}{2(1-x)^2}$. Отже, при $0 \leq x < 1$ функція $f(x)$ є зростаючою, тому найменшим її значенням в цій множині є $f(0) = 0$. ∎

Наступну нерівність часто називають також нерівністю Чебишова.

НЕРІВНІСТЬ В.7. (МАРКОВА) *Нехай випадкова величина Y набуває невід'ємні дійсні значення і $c > 0$. Тоді*

$$\mathbf{P}[Y \geq c] \leq \frac{\mathbf{E}[Y]}{c}.$$

ДОВЕДЕННЯ. Безпосередньо з означень випливає, що

$$\mathbf{E}[Y] \geq \mathbf{P}[Y \geq c] \cdot c + \mathbf{P}[Y < c] \cdot 0.$$

∎

Позначимо через X_i для $1 \leq i \leq n$ незалежні випадкові величини, кожна з яких набуває значення 1 із ймовірністю p, де $0 < p < 1$, і 0 із ймовірністю $q = 1 - p$. Сума $X = \sum_{i=1}^n X_i$ дорівнює кількості успіхів у n незалежних випробуваннях, при ймовірності p успішного проходження кожного з випробувань. Випадкова величина X має *біноміальний розподіл* або *розподіл Бернуллі*: $\mathbf{P}[X = k] = \binom{n}{k} p^k q^{n-k}$. Наступні нерівності оцінюють ймовірність відхилення величини X від свого середнього значення $\mathbf{E}[X] = pn$.

НЕРІВНІСТЬ В.8. (ОЦІНКА ЧЕРНОВА [83], 1952) *Для* $0 \le \epsilon < q$

$$\mathbf{P}[X \ge (p+\epsilon)n] \le \left[\left(\frac{p}{p+\epsilon}\right)^{p+\epsilon}\left(\frac{q}{q-\epsilon}\right)^{q-\epsilon}\right]^n.$$

ДОВЕДЕННЯ. Нехай параметр $t > 0$ є фіксованим дійсним числом, значення якого буде вибране нижче. Оскільки X_i незалежні, такими ж є випадкові величини e^{tX_i}. Використаємо той факт, що математичне сподівання добутку незалежних випадкових величин дорівнює добутку їх математичних сподівань. Отримаємо

$$\mathbf{E}\left[e^{tX}\right] = \mathbf{E}\left[\prod_{i=1}^{n} e^{tX_i}\right] = \prod_{i=1}^{n} \mathbf{E}\left[e^{tX_i}\right] = (q+pe^t)^n.$$

Оскільки функція $f(x) = e^{tx}$ монотонно зростає, подія $X \ge (p+\epsilon)n$ еквівалентна тому, що $e^{tX} \ge e^{t(p+\epsilon)n}$. За нерівністю Маркова ймовірність останньої події не перевищує величини

$$\frac{\mathbf{E}\left[e^{tX}\right]}{e^{t(p+\epsilon)n}} = \left(\frac{q+pe^t}{e^{t(p+\epsilon)}}\right)^n.$$

Отже,

$$\mathbf{P}[X \ge (p+\epsilon)n] \le \left(\min_{u>1} \frac{q+pu}{u^{p+\epsilon}}\right)^n.$$

Похідна функції $g(u) = \frac{(q+pu)}{u^{p+\epsilon}}$ дорівнює $g'(u) = \frac{p(q-\epsilon)u - q(p+\epsilon)}{u^{1+p+\epsilon}}$. Елементарний аналіз показує, що в області $u > 1$ функція $g(u)$ досягає мінімуму при $u = \frac{q(p+\epsilon)}{p(q-\epsilon)}$ і її мінімальне значення дорівнює $\left(\frac{p}{p+\epsilon}\right)^{p+\epsilon}\left(\frac{q}{q-\epsilon}\right)^{q-\epsilon}$. Це й дає потрібну оцінку. ∎

Отриману оцінку можна зробити набагато зручнішою для користування ціною певного послаблення.

НЕРІВНІСТЬ В.9. (СПРОЩЕНИЙ ВАРІАНТ ОЦІНКИ ЧЕРНОВА)
При $0 \le \epsilon \le pq$

1) $\mathbf{P}[X \ge (p+\epsilon)n] \le e^{-\epsilon^2 n}$
2) $\mathbf{P}[X \le (p-\epsilon)n] \le e^{-\epsilon^2 n}$

ДОВЕДЕННЯ. Оцінимо зверху праву частину нерівності В.8. Для цього прологарифмуємо вираз у квадратних дужках, отримавши $-(p+\epsilon)\ln(1+\epsilon/p) - (q-\epsilon)\ln(1-\epsilon/q)$, і застосуємо до обох логарифмів нерівності В.5 та В.6 відповідно. Як бачимо, ця величина не перевищує

$$(p+\epsilon)\left(\frac{\epsilon^2}{2p^2} - \frac{\epsilon}{p}\right) + (q-\epsilon)\left(\frac{\epsilon}{q} + \frac{\epsilon^2/q^2}{2(1-\epsilon/q)}\right) =$$

$$\left(-\frac{1}{2pq} + \frac{\epsilon}{2p^2}\right)\epsilon^2 \le \left(-\frac{1}{2pq} + \frac{pq}{2p^2}\right)\epsilon^2 = \left(-\frac{1}{2} - \frac{1}{2q}\right)\epsilon^2 \le -\epsilon^2.$$

Це доводить нерівність 1.

Нерівність 2 зводиться до попередньої. Справді, перетворимо нерівність $X \le (p-\epsilon)n$ у $n - X \ge (q+\epsilon)n$. Звідси видно, що ймовірність виконання цієї події дорівнює ймовірності того, що $Y \ge (q+\epsilon)n$. Випадкова величина Y має тут такий же розподіл як і X, але з параметрами p і q, що помінялись місцями. Тепер можна застосувати оцінку 1, яка від p і q не залежить. ∎

В [69, 145] можна знайти інші варіанти оцінки Чернова, яким слід віддати перевагу у випадку $p = o(1)$ або $q = o(1)$ для зростаючого n. У тій же ситуації добре працює

НЕРІВНІСТЬ В.10. (БЕРНШТЕЙНА, 1911) *Для* $\epsilon > 0$,

$$\mathbf{P}[|X - pn| > \epsilon n] \le 2\exp\left\{-\frac{\epsilon^2 n}{2pq(1+\frac{\epsilon}{3pq})}\right\}.$$

ДОВЕДЕННЯ. Міститься в [9]. ∎

На завершення наведемо ще одну оцінку такого ж гатунку, достатню для застосувань у нашому курсі. Слідом за [139] ми дамо повністю "дискретне" доведення на випадок, якщо таке є бажаним з методичних міркувань. Елементарні доведення інших версій містяться в [149, 14].

НЕРІВНІСТЬ В.11. (ПРОСТИЙ ВАРІАНТ ОЦІНКИ ЧЕРНОВА) *Нехай* X *— кількість успіхів серед* n *незалежних випробувань, у кожному з яких ймовірність успіху дорівнює* $1/2 + \epsilon$, *де* $0 < \epsilon \le 1/2$. *Тоді*

$$\mathbf{P}\left[X \le \frac{n}{2}\right] \le \frac{1}{2}e^{-2\epsilon^2 n}.$$

ДОВЕДЕННЯ. Для $k \le n/2$ маємо

$$\begin{aligned}
\mathbf{P}[X = k] &= \binom{n}{k}(\tfrac{1}{2}+\epsilon)^k(\tfrac{1}{2}-\epsilon)^{n-k} \\
&\le \binom{n}{k}(\tfrac{1}{2}+\epsilon)^k(\tfrac{1}{2}-\epsilon)^{n-k}(\frac{1/2+\epsilon}{1/2-\epsilon})^{n/2-k} \\
&= \binom{n}{k}(\tfrac{1}{4}-\epsilon^2)^{n/2}.
\end{aligned}$$

Далі,

$$\mathbf{P}\left[X \le \frac{n}{2}\right] = \sum_{k=1}^{\lfloor n/2 \rfloor} \mathbf{P}[X = k] \le 2^{n-1}(1/4-\epsilon^2)^{n/2} = \tfrac{1}{2}(1-4\epsilon^2)^{n/2}.$$

Доведення завершується застосуванням нерівності В.4. ∎

Додаток Г.

Ключ до вправ

До розділу I

1.1. тбфхлфиуя **1.2.** опівночі **1.3.** a) абракадабра b) неперевершене c) припинити d) подорожчало **1.4.** a) ееоуаияпрхввтс b) рвтееоуаияпхвс **1.5.** ненадійно **1.6.** $33^{\lceil n/2 \rceil}$.

2.1. a) dqzosphqbogz b) university **2.2.** c) $3 \cdot 5 \cdot 7 \cdot \ldots \cdot 21 \cdot 23 \cdot 25$ **2.5.** ␣ - 1/7, а - 2/7, м - 2/7, л - 1/14, р - 1/14, у - 1/14 **2.6.** Престолонаступником проголошено Олелька Третього **2.7.** a) yekgacyagr b) polygram **2.8.** a) n^k b) 625 **2.9.** a) $(25!)^4$ b) Будь-які два ключі такі, що один з них можна отримати з другого одночасним переставленням рядків у обох верхніх або нижніх квадратах, або ж одночасним переставленням стовпчиків у обох лівих або правих квадратах. c) 625! d) $(25!/5!)^4$ **2.11.** a) исюіхахицюеюсюи b) снiжинки падають пухнасті **2.12.** г **2.15.** a) хябтцрфухибчрябу b) все довкола стало білим c) За тотожністю $-((-m)+k))+k=m$. **2.16.** a) исюінбесаусмеіьм b) снiжинки падають пухнасті **2.17.** Для $l \geq 1$ позначимо через U криптотекст без перших l букв, а через V — криптотекст без останніх l букв. Якщо l — довжина ключа, то обчислення різниці $U - V$ дає відкритий текст без перших l букв. Справжнє значення l знаходиться перебором, який припиняється в той момент, коли операція віднімання привела до змістовного тексту. **2.19.** a) oswhimriihuilroytetnskeotuthes b) and nothing new under the sun **2.20.** a) епшненеодврнноае b) університетський **2.22.** a) 2^8 b) 2^{18} c) $2^{k^2/2}$ **2.23.** a) орптхизнліаа b) перекотиполе **2.24.** какаду **2.25.** c) 0, якщо l непарне; $3 \cdot 5 \cdot 7 \cdot \ldots \cdot (l-3) \cdot (l-1)$, якщо l парне. **2.28.** За чинною інструкцією секретний виріб при виникненні будь-якої загрози знищується охоронцем.

3.1. a) 000000 010001 000000 010001 000000 010101 (ананас), 100000 000001 001111 010111 001110 010010 (яблуко) b) 110000 001111 010100 100011 010110 001011 c) 010000 001110 011011 110100 011000 011001 **3.2.** 2^n **3.3.** a) gagxa fgfav dxdfd fvxfd ggxdg xxagx gdxvx gxgfg xdgdv aaddg agggf fgvga b) nothing new under the sun **3.4.** $C = D_{K_1}(E_{K_2}(D_{K_3}(M)))$ **3.6.** Використати індукцію за i. **3.8.** $2^{2048} \cdot 32!$ **3.9.** $2^{4096} \prod_{i=1}^{64}(1-2^{-i})$ (див. теорему II.2.19) **3.10.** Зчеплення зашифрованих блоків: $M_i = D_K(C_i) \oplus C_{i-1}$. Зворотний зв'язок за виходом або криптотекстом: $M_i = C_i \oplus B_i$.

До розділу II

1.1. $E_2 \circ E_1 : (\mathcal{K}_2 \times \mathcal{K}_1) \times \mathcal{A}^* \to \mathcal{C}^*$ задається співвідношенням $E_2 \circ E_1(K_2, K_1, M) = E_2(K_2, E_1(K_1, M))$ для повідомлення M і ключа (K_2, K_1). **1.3.** Позначимо $N = n^l$. Розглядаємо E_K як перестановку N-елементної множини $\mathcal{A}^l$. Нехай вона розкладається на цикли з довжинами $x_1, \ldots, x_t$. Зокрема, $\sum_{i=1}^{t} x_i = N$. Зрозуміло, що $E_K^m = \mathrm{id}_{\mathcal{A}^l}$ для $m = \mathrm{НСК}(x_1, \ldots, x_t)$. Для a) оцінимо $m \leq \prod_{i=1}^{t} x_i$. За нерівністю між середнім геометричним і середнім арифметичним (нерівність B.3) маємо $m \leq (N/t)^t$. Для $t > 0$ права частина досягає максимуму $e^{N/e}$ при $t = N/e$. Для b) візьмемо оцінку $m \leq \mathrm{НСК}(2, 3, \ldots, N) \leq N^{\pi(N)}$.

2.1. a) 1 b) 8 **2.2.** 7 **2.3.** b) Див. доведення теореми Ламе в [32, розділ 4.5.3] або [3, розділ 2.2.2].

c) Скористаємось з формули $f_n = (\lambda_1^n - \lambda_2^n)/\sqrt{5}$, де $\lambda_1 = \lambda = \frac{1+\sqrt{5}}{2}$ і $\lambda_2 = \frac{1-\sqrt{5}}{2}$. З неї випливає, що $f_{n-1} < b \leq f_n$ для $n = \lceil \log_\lambda(\sqrt{5}b) \rceil$, і бажана оцінка є безпосереднім наслідком пунктів a і b. Її оптимальність випливає з пункту a.

Використану нами формулу для n-го члена послідовності Фібоначчі виведемо способом, викладеним в [30]. За означенням справедлива рівність $\begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} f_{n-1} \\ f_{n-2} \end{pmatrix}$, звідки випливає, що $\begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-2} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. Застосувавши відому з курсу лінійної алгебри процедуру діагоналізації матриці, отримаємо $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = A \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} A^{-1}$ для $A = \begin{pmatrix} \lambda_1 & \lambda_2/\sqrt{5} \\ 1 & 1/\sqrt{5} \end{pmatrix}$. Звідси $\begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix} = A \begin{pmatrix} \lambda_1^{n-2} & 0 \\ 0 & \lambda_2^{n-2} \end{pmatrix} A^{-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. З цієї рівності знаходимо, що $f_n = (\lambda_1^n - \lambda_2^n)/\sqrt{5}$. **2.6.** $u = 33$, $v = -40$ **2.7.** Індукцією за i. **2.11.** $2^4 \cdot 3^3 \cdot 11^2 \cdot 37$ **2.12.** 22 **2.13.** 14288 **2.14.** $\phi(1) = 1$, $\phi(2) = 1$, $\phi(3) = 2$, $\phi(4) = 2$, $\phi(5) = 4$, $\phi(6) = 2$, $\phi(7) = 6$, $\phi(8) = 4$, $\phi(9) = 6$, $\phi(10) = 4$, $\phi(11) = 10$, $\phi(12) = 4$, $\phi(13) = 12$, $\phi(14) = 6$, $\phi(15) = 8$, $\phi(16) = 8$, $\phi(17) = 16$, $\phi(18) = 6$, $\phi(19) = 18$,

$\phi(20) = 8$ **2.15.** 1, 2 **2.16.** 1, 2, 3, 4, 5, 6, 8, 10, 12 **2.17.** a) 157248 b) 633600 **2.18.** Вказати обернене відображення. **2.19.** $\phi_k(2)$ при $n = 2$; $2\phi_k(n)/\phi(n)$ при $n > 2$. При $n = 2$ кожна оборотня матриця є унімодулярною. При $n > 2$ кількість унімодулярних матриць можна знайти виходячи з таких міркувань. Оскільки спеціальна лінійна група $\mathrm{SL}_k(\mathbb{Z}_n)$ є ядром гомоморфізму $\det : \mathrm{GL}_k(\mathbb{Z}_n) \to \mathbb{Z}_n^*$, то за теоремою про гомоморфізм (див. напр. [29]) фактор-група $\mathrm{GL}_k(\mathbb{Z}_n)/\mathrm{SL}_k(\mathbb{Z}_n)$ ізоморфна із $\mathbb{Z}_n^*$, образом цього гомоморфізму. Тому $|\mathrm{SL}_k(\mathbb{Z}_n)| = |\mathrm{GL}_k(\mathbb{Z}_n)|/|\mathbb{Z}_n^*| = \phi_k(n)/\phi(n)$. Множина унімодулярних матриць є об'єднанням $\mathrm{SL}_k(\mathbb{Z}_n)$ з її суміжним класом — множиною матриць з визначником -1. (Або ж можна зауважити, що $\mathrm{SL}_k(\mathbb{Z}_n)$ є підгрупою індексу 2 у групі унімодулярних матриць.)

3.2. a) дк b) $a' = 25$, $s' = 24$, **ніколи**

3.3. a) Пропуск має номер 33, а літера Щ — 29. Звідси для дешифруючого ключа a' маємо співвідношення $29a' \equiv 33 \pmod{35}$. Домноживши ліву і праву частину на $29^{-1} \bmod 35 = 29$, отримаємо $a' = 12$. Записуємо криптотекст у числовій формі і після множення кожного елемента на 12 за модулем 35 дістаємо числовий еквівалент відкритого тексту ПИВО␣ЗАВЕЗЛИ.␣ВІДВАНТАЖУЙТЕ␣РАКІВ.

За дешифруючим ключем знаходимо шифруючий $a = 12^{-1} \bmod 35 = 3$. Множення на 3 за модулем 35 кожного елемента числового еквіваленту заданого повідомлення дає криптотекст ЄХОЕОЮЄЬЩЧАЄ␣МІЬИЬШРЯ

b) Літера О має номер 18, а Ф — 24. Звідси для дешифруючого ключа a' маємо співвідношення $24a' \equiv 18 \pmod{33}$. Ми не можемо як у попередньому пункті однозначно встановити a', бо НСД$(24, 33) = 3$, і число 24 за модулем 33 є необоротним. Однак про a' можна отримати досить багато інформації. Скоротимо обидві частини конгруенції і модуль на 3 (див. твердження 2.6). Маємо $8a' \equiv 6 \pmod{11}$. Домноживши ліву і праву частину на $8^{-1} \bmod 11 = 7$, отримаємо $a' \equiv 9 \pmod{11}$. Це дає для a' три можливості: $a' = 9$, 20, або 31. Першу з них слід відкинути відразу, оскільки 9 не взаємно просте з 33. Спроба дешифрувати заданий криптотекст за допомогою ключа 20 дає результат ЮОЗЖОЧЯШАЩДО, а за допомогою ключа 31 — РОЗПОЧИНАЄМО. Природно вважати, що правильним є останній варіант.

За дешифруючим ключем 31 знаходимо шифруючий $a = 31^{-1} \bmod 33 = 16$. З його допомогою перетворюємо заданий відкритий текст у криптотекст ЇАЯЬУПДИТЬ

c) Пропуск має номер 2, а літера Є — 10. Звідси для дешифруючого ключа a' отримуємо співвідношення $10a' \equiv 2 \pmod{36}$. Як і в попередньому пункті, ми не можемо звідси визначити a' однозначно, бо 10 і 36 не взаємно прості. Але тепер в нас є додаткова інформація. Оскільки літера О має номер 21, а Ь — 33, маємо ще одну конгруенцію $33a' \equiv 21 \pmod{36}$ (з неї самої теж неможливо однозначно знайти a'). Множники 10 і 33 взаємно прості.

Тому ми можемо застосувати розширений алгоритм Евкліда і знайти, що $10 \cdot 10 - 3 \cdot 33 = 1$. Звідси видно, що слід домножити першу конгруенцію на 10 і відняти від неї другу конгруенцію, домножену на 3. В результаті отримаємо $a' \equiv -43 \pmod{36}$, звідки $a' = 29$. Знаючи дешифруючий ключ, ми в стані розшифрувати заданий криптотекст: АЛІСО,␣Я␣ВТОМИВСЯ␣ЧЕКАТИ.␣БОБ.

За допомогою шифруючого ключа $a = 29^{-1} \bmod 36 = 5$ перетворюємо задане повідомлення у криптотекст СЇНПБОЄПЗЕВЄНЬНЕ.

3.4. a) Дешифруюче відображення переводить букви з номерами 23 і 18 у букви з номерами 18 і 17 відповідно. Тому для дешифруючого ключа a', s' маємо конгруенції $23a' + s' \equiv 18 \pmod{33}$ і $18a' + s' \equiv 17 \pmod{33}$. Віднявши від першої конгруенції другу, отримаємо $5a' \equiv 1 \pmod{33}$. Домноживши обидві частини на $5^{-1} \bmod 33 = 20$, визначимо $a' = 20$. Підставивши це значення в першу конгруенцію, знайдемо $s' = (18 - 23 \cdot 20) \bmod 33 = 20$. За допомогою дешифруючого ключа проводимо дешифрування заданого криптотексту і отримуємо повідомлення ВОГОНЬ. Далі знаходимо шифруючий ключ $a = (a')^{-1} \bmod 33 = 5$ і $s = (-as') \bmod 33 = 32$. З його допомогою проводимо шифрування заданого повідомлення і отримуємо криптотекст ІЯКЯДМИМ

b) $a' = 9$, $s' = 8$, TOBEORNOTTOBE, $a = 3$, $s = 2$, PSYKOEHASPE

3.5. c) $n\phi(n)$, 312, 660.

3.6. НСД$(a - 1, n)$, якщо s кратне НСД$(a - 1, n)$; і 0 інакше.

3.7. a) $\begin{pmatrix} 31 & 1 \\ 18 & 16 \end{pmatrix}$ b) ЦДИДЦП c) ЧҐІҐЧО d) A' з пункту a, $S' = \begin{pmatrix} 3 \\ 31 \end{pmatrix}$, ВЕРТАЙСЯ

3.8. a) Слову АЛІСА у відкритому тексті відповідає слово ␣ЄЦЖС у криптотексті. Звідси видно, що дешифруюче відображення має переводити біграми ЄЦ $= \begin{pmatrix} 7 \\ 26 \end{pmatrix}$ і ЖС $= \begin{pmatrix} 8 \\ 21 \end{pmatrix}$ в біграми ЛІ $= \begin{pmatrix} 15 \\ 11 \end{pmatrix}$ і СА $= \begin{pmatrix} 21 \\ 0 \end{pmatrix}$ відповідно. Таким чином, для дешифруючої матриці A' маємо співвідношення $A' \begin{pmatrix} 7 & 8 \\ 26 & 21 \end{pmatrix} = \begin{pmatrix} 15 & 21 \\ 11 & 0 \end{pmatrix}$, з якого знаходимо

$$A' = \begin{pmatrix} 15 & 21 \\ 11 & 0 \end{pmatrix} \begin{pmatrix} 7 & 8 \\ 26 & 21 \end{pmatrix}^{-1} = \begin{pmatrix} 15 & 21 \\ 11 & 0 \end{pmatrix} \begin{pmatrix} 3 & 28 \\ 6 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 33 \\ 33 & 2 \end{pmatrix}$$

(всі обрахунки за модулем 34). Числовим еквівалентом заданого криптотексту, розбитого на біграми, є

$$C = \begin{pmatrix} 4 & 11 & 3 & 20 & 11 & 8 & 16 & 27 & 10 & 32 & 7 & 8 \\ 22 & 28 & 4 & 31 & 05 & 21 & 0 & 05 & 14 & 33 & 26 & 21 \end{pmatrix}.$$

Знаючи дешифруючий ключ, отримуємо числовий еквівалент повідомлення

$$M = A'C = \begin{pmatrix} 16 & 17 & 33 & 23 & 6 & 21 & 16 & 22 & 30 & 33 & 15 & 21 \\ 6 & 11 & 05 & 08 & 33 & 0 & 18 & 17 & 18 & 0 & 11 & 0 \end{pmatrix},$$

тобто МЕНІ␣ДУЖЕ␣САМОТНЬО␣АЛІСА

Шифруючий ключ рівний $A = (A')^{-1} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$. З його допомогою шифруємо задане повідомлення, розбиття якого на біграми виглядає у цифровій формі як

$$M = \begin{pmatrix} 27 & 14 & 31 & 1 & 15 & 33 & 18 & 22 & 17 & 33 & 18 \\ 6 & 0 & 33 & 11 & 32 & 24 & 17 & 0 & 23 & 1 & 1 \end{pmatrix}.$$

Отримуємо

$$C = AM = \begin{pmatrix} 26 & 28 & 27 & 13 & 28 & 22 & 19 & 10 & 23 & 33 & 3 \\ 33 & 14 & 30 & 12 & 13 & 23 & 1 & 22 & 6 & 0 & 19 \end{pmatrix},$$

що є числовою формою криптотексту Ц␣ШКЧЬЙЇШЙТУПБИТУЕ␣АГП

b) З умови випливає, що дешифруюче відображення переводить біграми ДТ $= \begin{pmatrix} 5 \\ 22 \end{pmatrix}$ і ЛР $= \begin{pmatrix} 15 \\ 20 \end{pmatrix}$ в біграми БО $= \begin{pmatrix} 1 \\ 18 \end{pmatrix}$ і БЕ $= \begin{pmatrix} 1 \\ 6 \end{pmatrix}$ відповідно. Отже, для дешифруючої матриці A' маємо співвідношення $A' \begin{pmatrix} 5 & 15 \\ 22 & 20 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 18 & 6 \end{pmatrix}$. Ми не можемо звідси відразу, як у попередньому пункті, визначити A', бо матриця $\begin{pmatrix} 5 & 15 \\ 22 & 20 \end{pmatrix}$ має визначник $(-230) \bmod 34 = 8$, а тому не є оборотньою в $M_2(\mathbb{Z}_{34})$. Все ж постараємось витягнути з отриманої рівності максимум інформації. Позначимо через $\bar{A}'$ матрицю, яка отримується з матриці A' заміною кожного коефіцієнта його остачею від ділення на 17. Подібну операцію виконаємо над всіма матрицями із рівності. Рівність при цьому не порушиться (операція, яку ми виконуємо, є гомоморфізмом з кільця $M_2(\mathbb{Z}_{34})$ на кільце $M_2(\mathbb{Z}_{17})$). В результаті маємо $\bar{A}' \begin{pmatrix} 5 & 15 \\ 5 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 6 \end{pmatrix}$. Визначник матриці $\begin{pmatrix} 5 & 15 \\ 5 & 3 \end{pmatrix}$ дорівнює -60, числу взаємно простому з 17. Тож ця матриця оборотня в $M_2(\mathbb{Z}_{17})$, і ми отримуємо можливість обчислити $\bar{A}' = \begin{pmatrix} 1 & 1 \\ 1 & 6 \end{pmatrix} \begin{pmatrix} 5 & 15 \\ 5 & 3 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 1 \\ 1 & 6 \end{pmatrix} \begin{pmatrix} 11 & 13 \\ 10 & 7 \end{pmatrix} = \begin{pmatrix} 4 & 3 \\ 3 & 4 \end{pmatrix}$. Кожен із коефіцієнтів знайденої матриці $\bar{A}'$ відповідає якомусь із двох елементів в $\mathbb{Z}_{17}$, отже, для матриці A' маємо 8 можливостей. Точніше, $A' = \bar{A}' + 17B$, де $B \in M_2(\mathbb{Z}_2)$. Підставимо цей вираз для A' в нашу початкову рівність:

$$\left(\begin{pmatrix} 4 & 3 \\ 3 & 4 \end{pmatrix} + 17B \right) \begin{pmatrix} 5 & 15 \\ 22 & 20 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 18 & 6 \end{pmatrix}.$$

Останню рівність розглянемо тепер вже за модулем 2 (тобто замінюємо коефіцієнти кожної із матриць їх остачами від ділення на 2 — рівність збережеться). Отримаємо

$$\left(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + B \right) \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix},$$

звідки випливає, що $B \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$. Наслідком цього співвідношення для коефіцієнтів матриці $B = (b_{ij})$ є $b_{11} = b_{21} = 1$. Це скорочує кількість можливостей для матриці A' до 4.

Можна використати ще одну умову, а саме, оборотність матриці A'. З неї випливає, що $\det A'$ є взаємно простим з 34, а отже і з 2. Тому матриця, коефіцієнти якої є лишками відповідних коефіцієнтів матриці A' за модулем 2, мусить мати визначник 1. Пригадаємо, що йдеться про матрицю $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + B$. З врахуванням того, що $b_{11} = b_{21} = 1$, її визначник дорівнює b_{22}. Отже, $b_{11} = b_{21} = b_{22} = 1$, і для матриці A' залишається лише дві можливості $\begin{pmatrix} 4 & 3 \\ 3 & 4 \end{pmatrix} + 17 \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 21 & 3 \\ 20 & 21 \end{pmatrix}$ або $\begin{pmatrix} 4 & 3 \\ 3 & 4 \end{pmatrix} + 17 \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 21 & 20 \\ 20 & 21 \end{pmatrix}$. Спроба дешифрувати заданий криптотекст за допомогою першої матриці приводить до не цілком зрозумілого тексту БОБЕМЧЕКНЙМДУ␣ЗАВЖДИ . Дешифрування за допомогою другої матриці дає бездоганний результат БОБЕ␣ЧЕКАЙ␣ДЕ␣ЗАВЖДИ , отже, саме вона є дешифруючим ключем. Шифруючий ключ рівний $\begin{pmatrix} 21 & 20 \\ 20 & 21 \end{pmatrix}^{-1} = \begin{pmatrix} 3 & 2 \\ 2 & 3 \end{pmatrix}$. З його допомогою задане повідомлення перетворюється у криптотекст ХҐЖШУХХБЄФІВРПЯИЩБ␣БТД

3.9. a) $A' = \begin{pmatrix} 1 & 31 \\ 31 & 5 \end{pmatrix}$, СТІЙНАСВОЄМУ b) СЛАВА␣КПСС c) $A' = \begin{pmatrix} 3 & 21 \\ 25 & 2 \end{pmatrix}$, НЕТООКТНЕМАР

3.10. Див. розв'язання пункту b вправи 3.8. Відповіді: a) $A' = \begin{pmatrix} 5 & 28 \\ 23 & 3 \end{pmatrix}$, LET'S␣GO␣TO␣THE␣CINEMA,␣BOB. b) $A' = \begin{pmatrix} 1 & 10 \\ 29 & 7 \end{pmatrix}$, ВІТАННЯВСІМІСАЄВ

3.11. a) Виходимо з того, що дешифруюче відображення переводить

біграми ІТ $= \begin{pmatrix} 11 \\ 22 \end{pmatrix}$, ьГ $= \begin{pmatrix} 30 \\ 3 \end{pmatrix}$, КЇ $= \begin{pmatrix} 14 \\ 12 \end{pmatrix}$ у біграми АТ $= \begin{pmatrix} 0 \\ 22 \end{pmatrix}$, АЛ $= \begin{pmatrix} 0 \\ 15 \end{pmatrix}$, КА $= \begin{pmatrix} 14 \\ 0 \end{pmatrix}$ відповідно. Зокрема, для дешифруючої матриці A' маємо в $\mathrm{M}_2(\mathbb{Z}_{33})$ співвідношення $A' \begin{pmatrix} 11 & 14 \\ 22 & 12 \end{pmatrix} = \begin{pmatrix} 0 & 14 \\ 22 & 0 \end{pmatrix}$ і $A' \begin{pmatrix} 30 & 14 \\ 3 & 12 \end{pmatrix} = \begin{pmatrix} 0 & 14 \\ 15 & 0 \end{pmatrix}$. Визначники матриць, що входять у першу рівність, мають з 33 НСД рівний 11, а визначники матриць з другої рівності — 3. Тому жодної з рівностей самої по собі недостатньо, щоб визначити A'. Розглянемо першу рівність як співвідношення в $\mathrm{M}_2(\mathbb{Z}_3)$, а другу — в $\mathrm{M}_2(\mathbb{Z}_{11})$. Отримуємо $A'_1 \begin{pmatrix} 2 & 2 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}$ і $A'_2 \begin{pmatrix} 8 & 3 \\ 3 & 11 \end{pmatrix} = \begin{pmatrix} 0 & 3 \\ 4 & 0 \end{pmatrix}$, де пара матриць A'_1 і A'_2 є образом матриці A' відносно гомоморфізму, описаного у твердженні 2.16. Звідси знаходимо

$$A'_1 = \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 2 & 2 \\ 1 & 0 \end{pmatrix}^{-1} = \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 2 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

і

$$A'_2 = \begin{pmatrix} 0 & 3 \\ 4 & 0 \end{pmatrix} \begin{pmatrix} 8 & 3 \\ 3 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 0 & 3 \\ 4 & 0 \end{pmatrix} \begin{pmatrix} 10 & 3 \\ 3 & 3 \end{pmatrix} = \begin{pmatrix} 9 & 9 \\ 7 & 1 \end{pmatrix}.$$

Маючи A'_1 та A'_2 і використовуючи Китайську теорему про остачі для кожного з чотирьох коефіцієнтів (див. приклад 2.11), знаходимо $A' = \begin{pmatrix} 31 & 31 \\ 18 & 1 \end{pmatrix}$. Результатом дешифрування є АЯДІВЧИНАПОЛТАВКАНАТАЛКА

b) $A' = \begin{pmatrix} 17 & 33 \\ 1 & 2 \end{pmatrix}$. ДОЛІ␣НЕ␣УНИКНУТИ␣ВАНГА

3.12. $A' = \begin{pmatrix} 1 & 33 \\ 0 & 1 \end{pmatrix}$, $S' = \begin{pmatrix} 1 \\ 31 \end{pmatrix}$, ЖДИ␣МЕНЕ␣ПІД␣ВЕРБОЮ␣НАДВЕЧІР (див. підпункт *"Атака з відомим відкритим текстом"* пункту 3.3).

3.13. $A' = \begin{pmatrix} 17 & 16 & 17 \\ 17 & 17 & 16 \\ 16 & 17 & 17 \end{pmatrix}$, ПРИШЛІТЬТРОЯНДИІІКРУДЖЕЙМСБОНД

3.14. a) є безпосереднім наслідком асоціативності множення матриць. b) випливає з a.

3.15. c) $n^k \phi_k(n)$, 106 299 648, 689 990 400.

3.16. a) E є гомоморфізмом з адитивної групи $\mathbb{Z}_n^k$ на себе. Тому всі біграми з $E(\mathbb{Z}_n^k)$ мають однакову кількість прообразів. Ця кількість більша від 1, бо інакше E було б ін'єктивним, що із врахуванням теореми про обернене відображення суперечило б твердженню 3.5.

b) безпосередньо випливає з a.

3.17. a) НСД $(\det(A - I_2), n) = 1$ b) Якщо n просте, біграми утворюють 2-вимірний лінійний простір над полем $\mathbb{Z}_n$. Легко перевірити, що нерухомі біграми утворюють власний лінійний підпростір. Він може бути або 0-вимірним — єдина біграма aa, або 1-вимірним — рівно n біграм.

3.18. a) За умовою, $A^2 = I_k$. За мультиплікативністю визначника звідси маємо співвідношення $(\det A)^2 = 1$. В полі $\mathbb{Z}_p^*$ многочлен $x^2 - 1$ має не більше двох коренів, а саме, 1 або -1.

b) 4 при $p = 2$ і 2 при $p > 2$. *Розв'язання:* Слід знайти кількість матриць $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, коефіцієнти яких задовольняють умови $b(a+d) = c(a+d) = 0$ і $a^2 + bc = d^2 + bc = ad - bc = 1$. Розглянемо два випадки. Перший полягає в тому, що $a + d \neq 0$. Тоді умови на коефіцієнти зводяться до таких: $b = c = 0$ і $a^2 = d^2 = ad = 1$, звідки $a = d = \pm 1$. Зауважимо, що при $p = 2$ остання рівність суперечить умові $a + d \neq 0$. Отже, у першому випадку маємо 2 матриці при $p > 2$ і жодної при $p = 2$. У другому випадку, коли $a + d = 0$, з умов на коефіцієнти випливає рівність $a^2 + bc = -1$, що дає суперечність при $p > 2$. При $p = 2$ отримуємо $a = d = 0$, $b = c = 1$, або $a = d = 1$, $bc = 0$ — всього 4 матриці.

c) $p(p+1)$. *Розв'язання:* На коефіцієнти матриці A маємо умови $a^2 + bc = d^2 + bc = 1$, $b(a + d) = c(a + d) = 0$, $ad - bc = -1$. Додавши останню рівність до чотирьох перших, отримаємо $a(a + d) = b(a + d) = c(a + d) = d(a + d) = 0$. Звідси $a + d = 0$, бо інакше було б $a = b = c = d = 0$. Тому умови на коефіцієнти зводяться до наступних двох: $d = -a$ і $bc = 1 - a^2$. При $a = \pm 1$ один із коефіцієнтів b або c повинен дорівнювати 0, а інший може бути довільним — всього $2(2p-1)$ матриць. При $a \neq \pm 1$ кожне $b \neq 0$ визначає єдине c — всього $(p-2)(p-1)$ матриць.

d) 1876. *Розв'язання:* Для матриці $A \in \mathrm{M}_2(\mathbb{Z}_{33})^*$ нехай $A_1 \in \mathrm{M}_2(\mathbb{Z}_3)^*$ і $A_2 \in \mathrm{M}_2(\mathbb{Z}_{11})^*$ — образи матриці A відносно ізоморфізму з наслідку 2.17. A є матрицею інволюції тоді і тільки тоді, коли і A_1, і A_2 є матрицями інволюцій. Отже, кількість матриць інволюцій в $\mathrm{M}_2(\mathbb{Z}_{33})^*$ дорівнює добутку кількостей матриць інволюцій в $\mathrm{M}_2(\mathbb{Z}_3)^*$ і $\mathrm{M}_2(\mathbb{Z}_{11})^*$, кожна з яких обчислюється за пунктами a, b і c.

e) 736. Див. попередній пункт.

f) $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 14 & 13 \\ 13 & 14 \end{pmatrix}, \begin{pmatrix} 1 & 13 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 13 & 1 \end{pmatrix}, \begin{pmatrix} 12 & 13 \\ 13 & 12 \end{pmatrix}, \begin{pmatrix} 25 & 0 \\ 0 & 25 \end{pmatrix}, \begin{pmatrix} 25 & 13 \\ 0 & 25 \end{pmatrix}, \begin{pmatrix} 25 & 0 \\ 13 & 25 \end{pmatrix}$.

Розв'язання: Для матриці $A \in \mathrm{M}_2(\mathbb{Z}_{26})^*$ нехай $A_1 \in \mathrm{M}_2(\mathbb{Z}_2)^*$ і $A_2 \in \mathrm{M}_2(\mathbb{Z}_{13})^*$ — образи матриці A відносно ізоморфізму з наслідку 2.17. Нескладно побачити, що A є матрицею інволюції з визначником 1 тоді і тільки тоді, коли

і A_1, і A_2 є матрицями інволюцій з визначником 1. Для A_1 є 4 можливості (див. пункт b): $\begin{pmatrix}1&0\\0&1\end{pmatrix}$, $\begin{pmatrix}0&1\\1&0\end{pmatrix}$, $\begin{pmatrix}1&1\\0&1\end{pmatrix}$, $\begin{pmatrix}1&0\\1&1\end{pmatrix}$. Для A_2 є 2 можливості: $\begin{pmatrix}1&0\\0&1\end{pmatrix}$ і $\begin{pmatrix}-1&0\\0&-1\end{pmatrix}$. Для кожної із 8 можливих пар A_1, A_2 відповідну матрицю A утворюємо так: кожен коефіцієнт матриці A отримується із відповідних коефіцієнтів матриць A_1 та A_2 за допомогою Китайської теореми про остачі (а точніше, за допомогою алгоритму, викладеного в її доведенні).

3.19. a) Якщо σ переводить i-ту букву k-грами у j-у, то j-ий рядок матриці A має 1 на i-тому місці і 0 на всіх інших.

b) $\begin{pmatrix}0&0&0&0&1\\0&0&0&1&0\\1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&0\end{pmatrix}$

3.20. a) Ця ймовірність дорівнює кількості матриць в $\mathrm{M}_k(\mathbb{Z}_2)^*$ поділеній на загальну кількість матриць в $\mathrm{M}_k(\mathbb{Z}_2)$, тобто $\phi_k(2)/2^{k^2}$. За теоремою 2.19 це число дорівнює $\prod_{i=1}^{k}(1-(1/2)^i)$. Для всіх k оцінкою знизу цієї величини буде добуток $\prod_{i=1}^{\infty}(1-(1/2)^i)$. Використаємо нерівність $1-y \geq 4^{-y}$, справедливу для $0 \leq y \leq 1/2$ і рівність $\sum_{i=1}^{\infty} x^i = x/(1-x)$, справедливу для $-1 < x < 1$. В результаті для $0 \leq x \leq 1/2$ отримуємо, $\prod_{i=1}^{\infty}(1-x^k) \geq 4^{\sum_{i=1}^{+\infty} -x^k} = 4^{-x/(1-x)}$. При $x = 1/2$ маємо потрібну оцінку.

b) $\sum_{i=1}^{\infty}(\frac{3}{4})^{i-1}\frac{1}{4}i = 4$ c) $1-(3/4)^{10} > 0.94$.

До розділу III

2.1. a) Адитивна складність n, мультиплікативна $n(n+1)/2$. b) Адитивна n, мультиплікативна n. Використати рівність

$$f(x, x_0, x_1, \ldots, x_n) = (\ldots((x_n x + x_{n-1})x + x_{n-2})x + \cdots + x_1)x + x_0.$$

2.2. Припустимо, що f обчислюється деякою прямолінійною програмою. Тоді f можна задати як поліном із цілими коефіцієнтами від змінних x_1 і x_2. Розглянемо $f(x_1, 2) - 1$ як поліном від змінної x_1 над $\mathbb{R}$. Він набуває значення 0 на нескінченній множині (непарних цілих чисел), що дає суперечність. **2.5.** $z_1 = (x_1 + x_2)y_1$, $z_2 = x_1(y_2 - y_1)$, $z_3 = x_2(y_1 + y_2)$, $f_1 = z_1 - z_3$, $f_2 = z_1 + z_2$. **2.6.** a) $z_1 = (x_{12} - x_{22})(y_{21} + y_{22})$, $z_2 = (x_{11} + x_{22})(y_{11} + y_{22})$, $z_3 = (x_{12} - x_{21})(y_{21} + y_{12})$, $z_4 = (x_{11} + x_{12})y_{22}$, $z_5 = x_{11}(y_{12} - y_{22})$, $z_6 = x_{22}(y_{21} - y_{11})$, $z_7 = (x_{21} + x_{22})y_{11}$, $f_{11} = z_1 + z_2 - z_4 + z_6$, $f_{12} = z_4 + z_5$, $f_{21} = z_6 + z_7$, $f_{22} = z_2 - z_3 + z_5 - z_7$. b) Використати твердження Б.3. c) Використати обидва попередні пункти. **2.7.** a) Нижня оцінка випливає з того, що серед програм довжини l найбільшу константу обчислює програма $z_1 = 1 + 1$, $z_i = z_{i-1} \cdot z_{i-1}$ для $1 < i \leq l$. Цей факт доводиться індукцією за l. Для доведення верхньої оцінки запишемо x у двійковій системі $x = x_l 2^l + \ldots + x_1 2 + x_0$, де $x_i \in \{0, 1\}$. Дивлячись на цей вираз як на поліном степеня l, в який замість змінної підставлено 2, проведемо обчислення за схемою Горнера.

b) Використаємо формулу $f_n = (\lambda_1^n - \lambda_2^n)/\sqrt{5}$, де $\lambda_1 = \frac{1+\sqrt{5}}{2}$ і $\lambda_2 = \frac{1-\sqrt{5}}{2}$, виведену при розв'язанні вправи II.2.3. З врахуванням того, що $0 < \lambda_2 < 1$, з цієї формули випливає, що верхня оцінка з пункту a лише на константу відрізняється від $2(\log_2 \lambda_1)n$.

c) Виходимо із встановленої при розв'язанні вправи II.2.3 рівності $\begin{pmatrix}f_n\\f_{n-1}\end{pmatrix} = \begin{pmatrix}1&1\\1&0\end{pmatrix}^{n-2}\begin{pmatrix}1\\1\end{pmatrix}$. Для піднесення матриці до степеня використовуємо бінарний метод. Це займе менше $2\log_2 n$ множень матриць розміру 2×2, причому при кожному такому множенні відбуваються 8 множень і 4 додавання цілих чисел. **2.10.** a) $z_1 = x_1 \oplus x_4$, $z_2 = x_2 \oplus x_5$, $z_3 = x_3 \oplus x_6$, $z_4 = z_1 \wedge z_2$, $f = z_3 \wedge z_4$. **2.11.** a) $u_0 = x_0 \oplus y_0$, $z_1 = x_0 y_0$, $z_2 = z_1 \oplus x_1$, $u_1 = z_2 \oplus y_1$, $z_3 = x_1 \wedge y_1$, $z_4 = z_1 z_3$, $z_5 = x_1 y_1$, $u_2 = z_4 \wedge z_5$.

3.1. Використати оцінку Чернова з додатку В. **3.5.** a) Індукція за довжиною виразу. b) Індукція за m. Випадок $m = 1$ випливає із твердження Б.7. Припустимо, наше твердження справедливе для $m - 1$. Поліном $p(x_1, \ldots, x_m)$ степеня d можна подати у вигляді

$$\sum_{i=0}^{k} x_1^i \cdot p_i(x_2, \ldots, x_m), \tag{1}$$

де $k \leq d$, а p_i є деяким поліномом від $m-1$ змінної степеня щонайбільше $d - i$. Поліном p_k є ненульовим. За індуктивним припущенням він набуває ненульові значення на принаймні $h^{m-1} - (d-k)h^{m-2}$ наборах з H^{m-1}. Для кожного такого набору вираз (1) є поліномом степеня k від змінної x_1, який не є нулем для щонайменше $h - k$ значень змінної. Отже, поліном p набуває ненульові значення на щонайменше $(h^{m-1} - (d-k)h^{m-2})(h-k) \geq h^m - dh^{m-1}$ наборах з H^m. c) Однобічна помилка $1/n$. Використати попередні пункти. **3.6.** b) $\|Y\|/\|X\|$.

4.1. Послідовність (a_i, b_i) є такою: 1,35; 1,18; 1,10; 1,6; 4,6; 4,5. Вихід: 5.

До розділу IV

1.1. 2 для $p = 3$; 2, 3 для $p = 5$; 3, 5 для $p = 7$; 2, 6, 7, 8 для $p = 11$. **1.2.** a) По 1 елементові 1-го і 2-го порядків, по 10 елементів 11-го і 22-го порядків. b) За теоремою Лагранжа можливими порядками є числа 1, 2, q, і $2q$. В кожній групі одиниця є єдиним елементом 1-го порядку. $p-1$ є єдиним елементом 2-

го порядку, бо рівняння $X^2 = 1$ має в полі $\mathbb{Z}_p^*$ два розв'язки 1 і -1. Елементи порядку $2q$ є первісними коренями і їх є $\phi(p-1) = q-1$. Залишаються $q-1$ елементів, які мають порядок q. **1.5.** a) Використати попередню задачу і доведення твердження Б.1. b) $2^5 \bmod 13 = 6$, $2^7 \bmod 13 = 11$, $2^{11} \bmod 13 = 7$. **1.7.** a) 1 b) 1 **1.8.** Показати, що затрачається не більше кроків, ніж алгоритмом Евкліда при обчисленні НСД(x,n), і скористатися оцінкою ефективності останнього. **1.9.** Використати еквівалентність умов 1 і 3 із леми 1.5. **1.10.** Використати наслідок II.2.13 і пункт b попередньої задачі. **1.11.** b) випливає з пункту a за таким загальним фактом: якщо є гомоморфізм з однієї скінченної групи на іншу, то кожен елемент із образу має стільки прообразів, скільки елементів міститься в ядрі гомоморфізму. Для кожного конкретного гомоморфізму цей факт можна (і варто) довести безпосередньо. **1.12.** За пунктом a задачі 1.9. **1.13.** a) За мультиплікативністю символу Лежандра або за еквівалентністю умов 1 і 3 із леми 1.5. b) Використати попередній пункт і вказати обернене відображення $f_{z'}$ за допомогою пункту d вправи 1.12.

1.15. a) За мультиплікативністю символу Якобі $\left(\frac{x}{n}\right) = \left(\frac{x}{p}\right)\left(\frac{x}{q}\right)$, тому $\left(\frac{x}{n}\right) = 1$ для тих і лише тих x, для яких або 1) $\left(\frac{x_1}{p}\right) = \left(\frac{x_2}{q}\right) = 1$, або 2) $\left(\frac{x_1}{p}\right) = \left(\frac{x_2}{q}\right) = -1$. Тут $x_1 = x \bmod p$ і $x_2 = x \bmod q$. За наслідком 2.13 з Китайської теореми про остачі, між елементами $x \in \mathbb{Z}_n^*$ і парами $x_1 \in \mathbb{Z}_p^*$, $x_2 \in \mathbb{Z}_q^*$ існує взаємно однозначна відповідність. З означення символу Лежандра і твердження b вправи 1.9 випливає, що пар з властивостями 1 і 2 однакова кількість. Залишається зауважити, що пари з властивістю 1 і лише вони відповідають квадратичним лишкам $x \in \mathcal{Q}_n$.

b) Ін'єктивність відображення f_z випливає з того, що до нього є обернене відображення $f_{z^{-1} \bmod n}$. Нескладно зауважити, що $f_z(x) \in \widetilde{\mathcal{Q}}_n$ для $x \in \mathcal{Q}_n$ і $z \in \widetilde{\mathcal{Q}}_n$. Звідси і з пункту a випливає сюр'єктивність.

1.16. a) випливає з критерію Ойлера. b) є наслідком попереднього пункту. c) Нехай $q_1, \ldots, q_s$ — всі прості співмножники із розкладу числа p (деякі з них можуть зустрічатися в цьому списку кілька разів). За пунктом a і означенням символу Якобі маємо

$$\left(\frac{-1}{p}\right) = \prod_{i=1}^{s}\left(\frac{-1}{q_i}\right) = (-1)^{\sum_{i=1}^{s}(q_i-1)/2}.$$

Для завершення доведення досить зауважити, що

$$\frac{p-1}{2} = \frac{1}{2}\left(\prod_{i=1}^{s} q_i - 1\right) = \frac{1}{2}\left(\prod_{i=1}^{s}(1+2\frac{q_i-1}{2}) - 1\right) = 2t + \sum_{i=1}^{s}\frac{q_i-1}{2}$$

для деякого цілого t. **1.18.** У зведеннях між задачами використати бінарний пошук.

2.1. a) так b) ні **2.2.** a) 4, 11 b) 8, 13 **2.3.** a) 217 b) 341. **2.4.** Спершу припустимо, що n псевдопросте за основою x. Оскільки $n-1 = pq-1 \equiv q-1 \pmod{p-1}$, отримуємо $x^{q-1} \equiv 1 \pmod{p}$. З іншого боку, $x^{p-1} \equiv 1 \pmod{p}$ за малою теоремою Ферма. Оскільки d є лінійною комбінацією чисел $p-1$ і $q-1$, то маємо $x^d \equiv 1 \pmod{p}$. Симетричними міркуваннями отримуємо $x^d \equiv 1 \pmod{q}$, і таким чином $x^d \equiv 1 \pmod{n}$. Легко проводяться і зворотні міркування. Кількість основ в $\mathbb{Z}_n^*$ дорівнює d^2 (використати Китайську теорему про остачі і те, що конгруенція $x^d \equiv 1 \pmod{p}$ при $d \mid p-1$ має рівно d розв'язків). **2.7.** a) Використати тотожність $x^k - 1 = (x-1)(x^{k-1} + \ldots + x + 1)$. b) Використати тотожність $x^k + 1 = (x+1)(x^{k-1} - x^{k-2} + \ldots - x + 1)$ для непарного k. **2.8.** a) Числа $\geq n$ послідовно тестуються на простоту, поки не буде знайдене перше просте. Тестування проводиться за допомогою детермінованого тесту із пункту 2.6. b) Вибираємо випадкове число між n і $2n$, і тестуємо його на простоту за допомогою k_1-кратного тесту Соловея-Штрассена. Якщо це число проходить тест, то подаємо його на вихід, якщо ні, то вибираємо для такого ж тестування нове число. Якщо просте число таким чином не знайдене за $2k_2 \ln n$ спроби, припиняємо роботу без жодного результату.

Зрозуміло, що цей алгоритм є поліноміальним для довільних констант k_1 і k_2. Для будь-якого $\epsilon \in (0,1)$ ці константи можна вибрати досить великими для того, щоб алгоритм знаходив просте число з ймовірністю принаймні $1-\epsilon$. Справді, ймовірність того, що алгоритм подасть на вихід складене число, не перевищує ймовірності помилки тесту Соловея-Штрассена, тобто 2^{-k_1}. Інша несприятлива можливість полягає в тому, що алгоритм закінчить роботу безуспішно, тобто кожне із вибраних випадково і незалежно $2k_2 \ln n$ чисел виявиться складеним. Ймовірність останньої події дорівнює $(1 - \frac{\pi(2n)-\pi(n)}{n})^{2k_2 \ln n}$. Це не перевищує e^{-k_2}, що можна довести з використанням оцінок на $\pi(n)$ із пункту 1.3.

3.1. b) Припустимо, що $q = p + 2d$. Тоді $n + d^2 = (p+d)^2$. Звідси такий алгоритм. Послідовно перебираємо числа $t^2 > n$, і перевіряємо, чи $t^2 - n$ є квадратом. Якщо так, то маємо $p = t - d$, $q = t + d$. **3.2.** a) x дає таке $c \neq \pm 1$, що $c^2 \equiv 1 \pmod{n}$. Тоді НСД$(c+1, n)$ є нетривіальним дільником числа n. b) Слід вибирати p і q такими, щоб НСД$(p-1, q-1)$ був малим (див. вправу 2.4).

4.1. Досить показати, що всі елементи із $\mathcal{Q}_n$ мають однакову кількість квадратних коренів в $\mathbb{Z}_n^*$. Далі див. вправи 1.9, 1.10 і 1.11. **4.2.** a) ні b) так **4.3.** Можна застосувати описану в цьому параграфі ймовірнісну процедуру. В даному випадку простіше скористатися пунктом 2 квадратичного закону взаємності, за яким 2 є нелишком. **4.4.** a) ± 9 b) ± 18 c) ± 24 **4.5.** 26, 51, 37, 40. **4.7.** (див. [99]) На вході $z \in \mathbb{J}_n$ алгоритм A' вибирає

випадковим чином число $r \in \mathbb{Z}_n^*$ і обчислює $x = r^2 z \bmod n$. Зауважимо, що при $z \in \mathcal{Q}_n$ число x є випадковим квадратичним лишком, а при $z \in \widetilde{\mathcal{Q}}_n$ — випадковим псевдоквадратом (вправи III.3.8 a та IV.1.15 b). Далі A' покладає рівноймовірно або $a = 1$, або $a = -1$ і обчислює $x' = ax \bmod n$. Оскільки n — ціле Блюма, то із вправи 1.16 та мультиплікативності символу Якобі випливає, що $-1 \in \widetilde{\mathcal{Q}}_n$. Тому x' є випадковим елементом множини $\mathbb{J}_n$. Якщо $x' \in \mathcal{Q}_n$ і $a = 1$, то $z \in \mathcal{Q}_n$; це ж справедливо, якщо $x' \in \widetilde{\mathcal{Q}}_n$ і $a = -1$; в інших випадках $z \in \widetilde{\mathcal{Q}}_n$. Що саме має місце, $x' \in \mathcal{Q}_n$ чи $x' \in \widetilde{\mathcal{Q}}_n$, визначається за допомогою алгоритму A. За припущенням, ймовірність помилки не перевищує $1/2 - 1/(\log n)^c$. Повторенням описаної процедури $t = \lceil (\log n)^{2c} \ln n \rceil$ разів і вибором відповіді, яка зустрічалась частіше, ймовірність помилки зменшується до $1/n$ (вправа III.3.1).

5.1. $9991 = 97 \cdot 103$.

6.1. Див. вправу 1.5. **6.2.** За пунктом 2 твердження 1.2, число g є первісним коренем за модулем $p = 2^m + 1$, де $m > 1$, тоді і лише тоді, коли $g^{2^{m-1}} \not\equiv 1 \pmod p$. За критерієм Ойлера остання умова рівносильна рівності $\left(\frac{g}{2^m+1}\right) = -1$. Для $g = 3$ за квадратичним законом взаємності маємо

$$\left(\frac{3}{2^m+1}\right) = \left(\frac{(2^m+1) \bmod 3}{3}\right) = \left(\frac{(-1)^m+1}{3}\right) = \left(\frac{2}{3}\right) = -1.$$

Тут використано, що m парне (інакше $2^m + 1$ ділилося б на 3).

7.2. a) 7 b) 10 c) 8.

До розділу V

2.1. a) $n = 3337$, $\phi(n) = 3220$, $d = 339$ b) 1620 1151 c) 1600 1518=МАЛО **2.2.** $n - \phi(n) = p + q - 1$ **2.3.** Див. вправу IV.3.1. **2.4.** a) Знайти цілі u і v такі, що $ue_1 + ve_2 = 1$, і обчислити $M = C_1^u C_2^v \bmod n$. b) Для знаходження таємного ключа за загальновідомим відкритим досить знати розклад на множники модуля n. Боб може обчислити $m = e_1 d_1 - 1$ і факторизувати n за допомогою процедури з пункту IV.5.2. **2.5.** a) За Китайською теоремою про остачі кількість таких x дорівнює кількості таких пар $(y, z) \in \mathbb{Z}_p \times \mathbb{Z}_q$, що $y^e \equiv y \pmod p$ і $z^e \equiv z \pmod q$, причому відповідні x та (y, z) пов'язані між собою співвідношеннями $y = x \bmod p$ і $z = x \bmod q$. Всі розв'язки, крім 0, конгруенції $y^e \equiv y \pmod p$ є також розв'язками рівняння $y^{e-1} = 1$ в $\mathbb{Z}_p^*$. Останніх є НСД$(e-1, p-1)$. Подібне має місце і для другої конгруенції. Отже, шукана кількість дорівнює (1+НСД$(e-1,p-1)$)(1+НСД$(e-1,q-1)$). b) 0, 1, 4, 5, 6, 9, 10, 11, 14. **2.6.** Нехай $ek \equiv 1 \pmod{\psi(n)}$, $1 \le k < \psi(n)$ (нагадаємо, що e і $\psi(n)$ взаємно прості). Тоді за твердженням 2.3 маємо $M^{ek} \bmod n = M$ для всіх повідомлень $M \in \mathbb{Z}_n$. **2.7.** Слід показати, що $E^{\phi(\psi(n))-1}(E(M)) = M$ для всіх повідомлень $M \in \mathbb{Z}_n$. Ліва частина дорівнює $M^{e^{\phi(\psi(n))}}$. За теоремою Ойлера $e^{\phi(\psi(n))} \equiv 1 \pmod{\psi(n)}$, і потрібна рівність випливає з твердження 2.3.

3.1. b) ОДИН **3.2.** Для початку зробимо таке зауваження. Розглянемо просте число $r = 4k + 3$ і припустимо, що $z \in \mathbb{Z}_r^*$. Тоді або $\left(\frac{z}{r}\right) = 1$, або $\left(\frac{-z}{r}\right) = 1$. Справді, якщо $\left(\frac{z}{r}\right) = -1$, то $\left(\frac{-z}{r}\right) = \left(\frac{-1}{r}\right)\left(\frac{z}{r}\right) = (-1)^{2k+1}(-1) = 1$. Тут використано мультиплікативність символу Лежандра і пункт a вправи IV.1.16.

Для $y \in \mathbb{Z}_n^*$, кореня з x, позначимо $y_1 = y \bmod p$ і $y_2 = y \bmod q$. Тоді чотири пари лишків $(\pm y_1, \pm y_2)$ визначають всі корені з x в $\mathbb{Z}_n^*$. Без втрати загальності можна припустити, що $\left(\frac{y_1}{p}\right) = \left(\frac{y_2}{q}\right) = 1$, адже для котрогось із чотирьох коренів це повинно виконуватись за попереднім зауваженням. Для такого $y = (y_1, y_2)$ маємо $b_1 = 1$, і нехай для нього $b_2 = b$. Тоді для $(-y_1, -y_2)$ маємо $b_1 = 1$ і $b_2 = 1 - b$. Остання рівність виконується, бо розглянуті два корені є взаємно протилежними за непарним модулем. Для інших двох коренів $(y_1, -y_2)$ і $(-y_1, y_2)$ маємо $b_1 = 0$ і знову різні b_2. **3.3.** Серед чотирьох коренів з $x \in \mathcal{Q}_n$ є рівно один в $\mathcal{Q}_n$, а саме з властивістю $\left(\frac{y_1}{p}\right) = \left(\frac{y_2}{q}\right) = 1$ — див. розв'язання попередньої вправи. **3.4.** Суперник вибирає випадково $M \in \mathbb{Z}_n^*$ і просить розшифрувати криптотекст $C = M^2 \bmod n$. З ймовірністю $1/2$ він отримує таке M', що $M \not\equiv \pm M' \pmod n$, і може факторизувати n (див. зведення факторизації до добування кореня із пункту IV.4.3).

4.2. b) Різним: А і Б. Втім, щоб відповісти на питання, не обов'язково проводити повне дешифрування. Можна міркувати так. Цілком очевидно, що другий біт першого відкритого тексту є 0, бо $64 = 8^2$. Очевидно також, що другий біт другого тексту і шостий біт першого однакові. Звідси випливає, що якби обидва відкриті тексти були ідентичними, то шостий біт другого тексту був би нулем, тобто $76 = -1 \bmod 77$ мало би бути квадратичним лишком за модулем 77. А це не так хоча б тому, що -1 не є квадратичним лишком за модулем 7.

5.1. b) 10 **5.2.** a) Так. Див. вправу IV.1.4. b) s c) Для тих a, для яких h теж є первісним елементом за модулем p. Це ті і лише ті a, які взаємно прості з $p - 1$ (див. вправу IV.1.5). Їх є $\phi(p-1)$. **5.3.** c) Перебираються всі можливі значення $h^r = (g^r)^a$, яких не більше, ніж s.

До розділу VI

1.1. Якщо $y = g^x \bmod p$, то $x \bmod 2 = 1$ тоді і лише тоді, коли $\left(\frac{y}{p}\right) = -1$ (див. лему IV.1.5), в той час як символ Лежандра обчислюється ефективно. **1.2.** Доведення проводимо від супротивного. Позначимо через $f : \mathcal{Q}_m \to \mathcal{Q}_m$ функцію Рабіна і припустимо, що для випадкового $z \in \mathcal{Q}_m$ поліноміальний ймовірнісний алгоритм A на вході $f(z)$ видає біт парності $z \bmod 2$ з ймовірністю щонайменше $1/2 + 1/n^c$, де ймовірність береться за випадковим вибором z і за випадковою послідовністю алгоритму. Нашою

метою є довести існування поліноміального ймовірнісного алгоритму для розпізнавання квадратичності за модулями, які є цілими Блюма. Спершу розглянемо поліноміальний ймовірнісний алгоритм $\hat{A}$. Отримавши на вхід x з $\mathbb{J}_m$, $\hat{A}$ обчислює $z = x^2 \bmod m$ і перевіряє умову $A(z) = x \bmod 2$. Якщо вона виконується, то $\hat{A}$ стверджує, що x є квадратичним лишком, а інакше — що x є псевдоквадратом. Зауважимо, що z є випадковим елементом множини $\mathcal{Q}_m$ і має рівно два квадратні корені в $\mathbb{J}_m$, які мають різну парність (вправа V.3.2). За припущенням, $A(z)$ дорівнює бітові парності кореня в $\mathcal{Q}_m$ з ймовірністю принаймні $1/2+1/n^c$. Отже, з цією ж ймовірністю алгоритм $\hat{A}$ правильно визначає, є випадковий елемент $x \in \mathbb{J}_m$ квадратичним лишком чи ні. Існування алгоритму для розпізнавання квадратичності випливає тепер із вправи IV.4.7. **1.3.** Звідності задач обчислення предикатів B_1 та B_2 одна до одної забезпечуються тотожностями $B_2(e,m,y) = B_1(e,m,y \cdot f_{e,m}(2))$ та $B_1(e,m,y) = B_2(e,m,y \cdot f_{e,m}(2^{-1}))$, де арифметичні операції виконуються в $\mathbb{Z}_m$ (зокрема, $2^{-1} = (m+1)/2$). Ці тотожності випливають із мультиплікативності функції RSA (за якою праві частини тотожностей дорівнюють $P(2x \bmod m)$ і $Q(2^{-1}x \bmod m)$, де $y = f_{e,m}(x)$), а також простого зауваження, що для $0 \le z < m$ нерівність $z < m/2$ еквівалентна парності числа $2z \bmod m$. **1.4.** За результатом попередньої задачі досить звести розкриття RSA до обчислення предикату B_2. За заданим криптотекстом $y = f_{e,m}(x)$ повідомлення x можна знайти, обчисливши послідовність бітів $B_2(e,m,y \cdot f_{e,m}(2^i)) = Q(2^i x \bmod m)$ для $i = 0,1,\ldots,\lfloor \log_2 m \rfloor$. Ця послідовність однозначно визначає x. Справді, $Q(x) = 0$ означає, що $0 \le x < \frac{m}{2}$; $Q(2x) = 0$ означає, що $0 \le x < \frac{m}{4}$ або $\frac{m}{2} < x < \frac{3}{4}m$; і т.д.

2.1. Див. [81] або [93]. **2.2.** Зіслатись на теорему III.2.2. **2.3.** Припустити супротивне і фіксацією деякої послідовності входів отримати із алгоритму послідовність схем поліноміального розміру, які б розрізняли випадкові і псевдовипадкові послідовності з імовірністю більшою, ніж $1/n^c$. **2.4.** Див. [150] або [81]. **2.5.** Припустивши, що деякий алгоритм вгадує попередній біт із суттєво кращою ніж 1/2 ймовірністю, показати, що генератор G не може бути псевдовипадковим. Це дасть суперечність із теоремою 2.7. **2.6.** Див. [79]. **2.8.** Використати теорему Ойлера. **2.9.** a) (100111 011001 100111, 85) b) HI

До розділу VII

2.1. Щоб отримати підпис Боба на документі M, можна попросити його підписати цифрові документи M_1 і M_2 такі, що $M = M_1 M_2 \bmod n_B$, і скористатися тим, що $D_B(M) = D_B(M_1)D_B(M_2) \bmod n_B$. **2.2.** Див. вправу V.3.4. **2.3.** За умовою, $f(x) = f(x')$ з ймовірністю щонайменше 1/2. Це дасть колізію лише якщо $x' \neq x$. На це не можна розраховувати у випадку, коли x є єдиним прообразом $f(x)$. Останнє може трапитись з імовірністю щонайбільше $2^{128}/2^{130} = 1/4$. Таким чином, з імовірністю щонайменше 1/4

виконується рівність $f(x) = f(x')$ і $f(x)$ має принаймні два прообрази. У припущенні, що саме це і має місце, нескладно обгрунтувати, що $x' \neq x$ щонайменше з імовірністю 1/2 (показати для кожного фіксованого x' і застосувати формулу повної ймовірності). **2.4.** a) $n/365$ (тут і в наступному пункті використати лінійність математичного сподівання) b) $n(n-1)/730$ c) Ймовірність того, що всі предмети різні, дорівнює $(1-1/n)(1-2/n)\ldots(1-\alpha\sqrt{n}/n)$. Застосувавши нерівність B.4 до кожного співмножника, отримаємо оцінку зверху у вигляді величини $e^{\alpha^2/2}$. d) Не менше 0.63 (застосувати попередній пункт). **2.5.** a) $2^{32}(2^{33}-1)(2^{k-64}-1)/(2^k-1)$ (тут і в наступному пункті використати лінійність математичного сподівання). Процедура полягає у випадковому виборі 2^{33} різних елементів з $\{0,1\}^k$, укладання таблиці значень функції на цих аргументах, і пошуці сортуванням аргументів з однаковими значеннями. b) 1 c) Попередньо зловмисник виготовляє 2^{32} незначні модифікації кожного із документів і знаходить серед них пару з одним і тим же вкороченням. Те, що ймовірність успішного знаходження вагома, випливає з аргументів попереднього пункту. (2^{32} модифікацій придумати зовсім неважко: досить вибрати в тексті 32 місця, де можна вставити — або не вставити зайвий пропуск, поставити крапку — або крапку з комою тощо.) **2.6.** Із рівності $h^q = 1$ в групі $\mathbb{Z}_p^*$ з врахуванням простоти q випливає, що порядок елемента h або 1 (тобто $h = 1$), або q. Зрозуміло, що за умови пункту a має місце другий із цих випадків. Пункт b можна переформулювати так. Нехай g фіксований первісний корінь за модулем p, а i випадкове ціле від 0 до $p-2$. Тоді з якою ймовірністю $g^{i(p-1)/q} \bmod p \neq 1$? Останнє питання рівносильне такому: з якою ймовірністю $i(p-1)/q \bmod (p-1) \neq 0$? Легко зауважити, що ця ймовірність дорівнює $1 - 1/q$.

3.1. ([136]) Аліна вибирає випадкового абонента в телефонному довіднику і покладає $a = 1$, якщо в його прізвищі непарна кількість літер, або $a = 0$ інакше. По телефону вона повідомляє Бобові телефонний номер вибраного абонента (залишаючи його прізвище у таємниці!). Боб називає свій випадковий біт b. Після цього Аліна називає свій біт a, а також прізвище абонента, щоб Боб міг перевірити отриманий раніше телефонний номер і тим самим переконатися, що біт a був справді вибраний наперед. Припускається, що за лічені секунди Боб не встигне знайти в довіднику абонента за його номером телефону. Також припускається, що в міській телефонній мережі абонентів з парною і непарною довжиною прізвища приблизно порівну. **3.5.** b) Див. твердження IV.4.4.

4.1. За мультиплікативністю символу Якобі $\left(\frac{M^e}{n}\right) = \left(\frac{M}{n}\right)^e = \left(\frac{M}{n}\right)$ (шифруюча експонента e завжди непарна). Тому обчисливши символи Якобі для надісланих Алісою криптотекстів, Боб зможе визначити, які з них відповідають тузам.

5.2. Скористатись твердженням Б.7.

До розділу VIII

1.4. Здогадом недетермінованого алгоритму є розклад числа n на прості співмножники. Далі алгоритм використовує критерій g із [13, § 4 розділу V]. **1.5.** Можна використати той факт, що число p є простим тоді і тільки тоді, коли деякий елемент g в $\mathbb{Z}_p^*$ має порядок $p-1$. Першим ділом недетермінований алгоритм вгадує такий елемент g. Щоб перевірити, що g справді має порядок $p-1$, алгоритм вгадує також розклад на прості співмножники $p-1 = q_1^{\alpha_1} q_2^{\alpha_2} \dots q_s^{\alpha_s}$ і перевіряє, що $g^{(p-1)/q_i} \not\equiv 1 \pmod p$ для $1 \le i \le s$. Алгоритм повинен також перевірити, що числа $q_1, q_2, \dots, q_s$ справді є простими. Щоб зробити це, він викликає сам себе рекурсивно для кожного q_i. **1.9.** Використати пункт 1 теореми III.4.2.

Бібліографія

Щороку проводяться міжнародні криптологічні конференції *CRYPTO, EUROCRYPT* та *AUSCRYPT*. Перша з них є центральною в галузі, друга організовується криптологами Європейського континенту, а третя відбувається в Австралії, Новій Зеландії або Сінгапурі. Матеріали цих конференцій регулярно видає *Springer Verlag* в серії *Lecture Notes in Computer Science, Advances in Cryptology*.

Найпрестижнішими заходами в теорії складності є *Annual ACM Symposium on Theory of Computing (STOC)* та *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, які традиційно включають секції з криптографії.

Власне криптографії присвячені періодичні видання *Journal of Cryptology* та *Cryptologia*. Огляд останніх досягнень регулярно робить журнал *SIGACT News* під рубрикою *Cryptography Column*.

[1] С. А. Абрамов. Некоторые оценки, связанные с алгоритмом Евклида. *Журнал вычислительной математики и математической физики*, 19:756–760, 1979.

[2] К. Айерлэнд, М. Роузен. *Классическое введение в современную теорию чисел*. М., Мир. 1987.

[3] А. Акритас. *Основы компьютерной алгебры с приложениями*. М., Мир. 1994.

[4] В. Б. Алексеев. Сложность умножения матриц. *Кибернетический сборник 25*. стр. 189–236. М., Мир. 1988.

[5] Н. Д. Андреев. *Статистико-комбинаторные методы в теоретическом и прикладном языковедении*. Л., Наука, ЛО. 1967.

[6] В. І. Андрійчук, Б. В. Забавський. *Алгебра і теорія чисел*. Університетська математика: основні курси. ВНТЛ, Львів, 1998, (готується до друку).

[7] А. Ахо, Дж. Хопкрофт, Дж. Ульман. *Построение и анализ вычислительных алгоритмов*. М., Мир. 1979.

[8] І. Я. Берегуляк. *Класичні методи криптування*. Львівський університет, 1997.

[9] С. Н. Бернштейн. *Теория вероятностей*. М.-Л., Гостехиздат, 1946.

[10] Э. Ф. Брикелл, Э. М. Одлижко. Криптоанализ: обзор новейших результатов. *Труды института инженеров электроники и радиофизики*, 76(5):75–94, 1988. (Малый тематический выпуск. Защита информации.)

[11] Б. Л. ван дер Варден. *Алгебра*. М., Наука, 2-е издание, 1979.

[12] О. Н. Василенко. Современные способы проверки простоты чисел. *Кибернетический сборник 25*, стр. 162–188. М., Мир. 1988.

[13] И. М. Виноградов. *Основы теории чисел*. М., Наука, 9-е издание, 1981. Попереднє видання вийшло в 1972 році.

[14] Я. Б. Воробець. Неопублікований рукопис. 1997.

[15] А. И. Галочкин, Ю. В. Нестеренко, А. Б. Шидловский. *Введение в теорию чисел*. М., изд. МГУ, 2-е издание, 1995.

[16] М. Гарднер. *От мозаик Пенроуза к надёжным шифрам*, главы 13, 14. М., Мир. 1993.

[17] Д. Ю. Григорьев. Нижние оценки в алгебраической сложности вычислений. *Записки научных семинаров ЛОМИ АН СССР, том 118*, стр. 25–82. Л., 1982.

[18] М. Гэри, Д. Джонсон. *Вычислительные машины и труднорешаемые задачи*. М., Мир. 1982.

[19] У. Диффи. Первые десять лет криптографии с открытым ключём. *Труды института инженеров электроники и радиофизики*, 76(5):54–74, 1988. (Малый тематический выпуск. Защита информации.)

[20] У. Диффи, М. Э. Хеллмэн. Новые направления в криптографии. *Труды института инженеров электроники и радиофизики*, 22:644–654, 1976.

[21] У. Диффи, М. Э. Хеллмэн. Защищённость и имитоустойчивость. Введение в криптографию. *Труды института инженеров электроники и радиофизики*, 67(3), 1979.

[22] А. К. Дойл. *Танцюючі чоловічки*.

[23] С. А. Дориченко, В. В. Яценко. *25 этюдов о шифрах*. Математические основы криптологии. М., ТЕИС, 1994.

[24] С. М. Ермаков. *Метод Монте-Карло и смежные вопросы*. М., Наука, 2-е издание, 1975.

[25] В. Жельников. *Криптография от папируса до компьютера*. ABF, Москва, 1996.

[26] М. И. Каргаполов, Ю. И. Мерзляков. *Основы теории групп*. М., Наука, 3-е издание, 1982.

[27] Р. М. Карп. Сводимость комбинаторных задач. *Кибернетический сборник 12*, 16–38. М., Мир. 1975.

[28] Н. Катленд. *Вычислимость. Введение в теорию вычислимых функций*. М., Мир. 1983.

[29] А. И. Кострикин. *Введение в алгебру*. М., Наука, 1977.

[30] А. И. Кострикин. *Введение в алгебру. Основы алгебры*. М., Наука, 1994.

[31] А. И. Кострикин, Ю. И. Манин. *Линейная алгебра и геометрия*. М., Наука, 2-е издание, 1986.

[32] Д. Кнут. *Искусство программирования для ЭВМ. Получисленные алгоритмы*, том 2. М., Мир. 1977. 2-ге видання оригіналу побачило світ у 1981 році у видавництві Addison-Wesley, Reading, MA, USA.

[33] Х. Коэн, Х. Ленстра, мл. Проверка чисел на простоту и суммы Якоби. *Кибернетический сборник 24*, стр. 101–146. М., Мир. 1987.

[34] С. А. Кук. Сложность процедур вывода теорем. *Кибернетический сборник 12*, 5–15. М., Мир. 1975.

[35] В. Н. Латышев. *Комбинаторная теория колец*. М., изд. МГУ, 1987.

[36] Л. А. Левин. Универсальные задачи перебора. *Проблемы передачи информации*, 9(3):115–116, 1973.

[37] С. Ленг. *Алгебра*. М., Мир. 1968.

[38] Х. У. Ленстра, мл. Алгоритмы проверки на простоту. В сб. *Алгебра и теория чисел с приложениями*, стр. 47–66. М., Мир. 1987.

[39] О. М. Макаров. *Введение в теорию оптимизации вычислений билинейных форм*. Наукова думка, Київ, 1983.

[40] А. И. Мальцев. *Алгоритмы и рекурсивные функции*. М., Наука, 2-е издание, 1986.

[41] К. Мандерс, Л. Эдлимэн. $\mathcal{NP}$-полные проблемы разрешения для квадратных уравнений с двумя неизвестными. *Кибернетический сборник 17*. М., Мир. 1980.

[42] Ю. И. Манин, А. А. Панчишкин *Введение в теорию чисел, Итоги науки и техники. Современные проблемы математики. Фундаментальные направления. Том 49. Теория чисел — 1*. М., ВИНИТИ, 1990.

[43] С. Мафтик. *Механизмы защиты в сетях ЭВМ*. М., Мир. 1993.

[44] Дж. Л. Месси. Введение в современную криптологию. *Труды института инженеров электроники и радиофизики*, 76(5):24–42, 1988. (Малый тематический выпуск. Защита информации.)

[45] Г. Л. Миллер. Гипотеза Римана и способы проверки простоты чисел. *Кибернетический сборник 23*, стр. 31–50. М., Мир. 1986.

[46] А. Э. Мильчин, ред. *Памятная книга редактора.* М., Книга, 2-е издание, 1988.

[47] Дж. Х. Мур. Несостоятельность протоколов криптосистем. *Труды института инженеров электроники и радиофизики*, 76(5):94–104, 1988. (Малый тематический выпуск. Защита информации.)

[48] Я. И. Перельман. *Живая математика.* М., Наука.

[49] Е. А. По. *Золотий жук.*

[50] Х. Роджерс. *Теория рекурсивных функций и эффективная вычислимость.* М., Мир. 1972.

[51] Г. Дж. Симмонс. Обзор методов аутентификации информации. *Труды института инженеров электроники и радиофизики*, 76(5):105–125, 1988. (Малый тематический выпуск. Защита информации.)

[52] Н. Дж. А. Слоун. Коды, исправляющие ошибки, и криптография. В Д. А. Кларнер, ред., *Математический цветник*, глава 6.3, стр. 432–472. М., Мир. 1983.

[53] М. Э. Смид, Д. К. Бранстед. Стандарт шифрования данных: прошлое и будущее. *Труды института инженеров электроники и радиофизики*, 76(5):43–54, 1988.

[54] Л. Стокмейер. Классификация вычислительной сложности проблем. *Кибернетический сборник 26*, стр. 20–83. М., Мир. 1989.

[55] Х. Уильямс. Проверка чисел на простоту с помощью вычислительных машин. *Кибернетический сборник 23*, стр. 51–99. М., Мир. 1986.

[56] А. М. Фаль. DES — этап в развитии криптологии. *Безопасность информации*, 2:18–21, 1995.

[57] А. М. Фаль. Алгоритм шифрования по ГОСТу 28147-89 и способы применения блочных шифров. *Безопасность информации*, 3:8–11, 1995.

[58] В. Феллер. *Введение в теорию вероятностей и её приложения*, том 1. М., Мир. 1984.

[59] А. Шамир, Р. Л. Райвест, Л. М. Адельман. Покер без карт. В сб. *Математический цветник*, под ред. Д. А. Кларнера, глава 1.5, стр. 58–66. М., Мир. 1983.

[60] К. Шеннон. Теория связи в секретных системах. В *Работы по теории информации и кибернетике*, стр. 333–402. М., Изд. иностр. лит., 1963.

[61] А. Шёнхаге, Ф. Штрассен. Быстрое умножение больших чисел. *Кибернетический сборник 10*, стр. 87–98. М., Мир. 1973.

[62] А. Шень. *Программирование: теоремы и задачи.* М., МЦНМО, 1995.

[63] Ф. Штрассен. Гауссова элиминация не оптимальна. *Кибернетический сборник 7*, стр. 67–70. М., Мир. 1970.

[64] L. M. Adleman. Algorithmic number theory — the complexity contribution. In *Proc. of the 35th IEEE Ann. Symp. on Foundations of Computer Science (FOCS)*, pages 88–113, 1994.

[65] L. M. Adleman and M.-D. Huang. Primality testing and two dimensional Abelian varieties over finite fields, volume 1512 of *Lecture Notes in Mathematics.* Springer Verlag, 1992.

[66] L. M. Adleman, C. Pomerance, and R. S. Rumely. On distinguishing prime numbers from composite numbers. *Ann. of Math.*, 117:173–206, 1983.

[67] W. B. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr. RSA and Rabin functions: certain parts are as hard as the whole. *SIAM Journal on Computing*, 17(2):194–209, 1988.

[68] W. R. Alford, A. Granville, and C. Pomerance. There are infinitely many Carmichael numbers. *Ann. of Math.*, 140:703–722, 1994.

[69] N. Alon and J. Spencer. *The probabilistic method.* John Wiley & Sons, New York, 1992.

[70] E. Bach. How to generate factored random numbers. *SIAM Journal on Computing*, 17:179–193, 1988.

[71] E. Bach and J. Shallit. *Algorithmic number theory*, volume 1. 1997.

[72] M. Bellare and S. Micali. How to sign given any trapdoor function. In *Proc. of the 20th ACM Ann. Symp. on Theory of Computing (STOC)*, pages 32–42, 1988.

[73] M. Ben-Or, B. Chor, and A. Shamir. On the cryptographic security of single RSA bits. In *Proc. of the 15th ACM Ann. Symp. on Theory of Computing (STOC)*, pages 421–430, 1983.

[74] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: how to remove intractability assumption. In *Proc. of the 20th ACM Ann. Symp. on the Theory of Computing (STOC)*, pages 113–131, 1988.

[75] E. Biham and A. Shamir. *Differential Cryptoanalysis of the Data Encryption Standard.* Springer Verlag, Berlin, 1993. (див. також матеріали конференцій CRYPTO'90–92.)

[76] M. Blum. Coin flipping by telephone: a protocol for solving impossible problems. In *Proc. 24th IEEE Spring Computer Conf. COMPCOM*, pages 133–137, 1982.

[77] L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383, 1986.

[78] M. Blum and S. Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In *CRYPTO'84 Proc.*, pages 289–299, Berlin, 1985. Springer Verlag. Lecture Notes in Computer Science, vol. 196.

[79] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–863, 1984.

[80] Bert den Boer. Diffie-Hellman is as strong as Discrete Log for certain primes. In *CRYPTO'88 Proc.*, pages 530–539. Springer Verlag, 1990. Lecture Notes in Computer Science, vol. 403.

[81] R. Boppana and R. Hirschfeld. Pseudorandom generators and complexity classes. In S. Micali, editor, *Randomness and Computation*, pages 1–26. JAI Press, Greenwich, 1989. Vol. 5, Adv. in Comput. Res.

[82] K. W. Campbell and M. J. Wiener. DES is not a group. In *CRYPTO'92*, pages 512–520, Berlin, 1993. Springer Verlag.

[83] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Math. Stat.*, 23:493–509, 1952.

[84] D. Coppersmith. Cheating at mental poker. In *Proc. CRYPTO'85*, pages 104–107, Berlin, 1986. Springer Verlag. Lecture Notes in Computer Science, vol. 218.

[85] J. M. DeLaurentis. A further weakness in the common modulus protocol for the RSA cryptosystem. *Cryptologia*, 8(3):253–259, 1984.

[86] W. Diffie and M. E. Hellman. Exhaustive cryptanalysis of the NBS data encryption standard. *Computer*, 10:74–84, June 1977.

[87] C. Dwork, U. Feige, J. Kilian, M. Naor, and M. Safra. Low communication 2-prover zero-knowledge proofs for $\mathcal{NP}$. In *CRYPTO'92*.

[88] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO'84 Proc.*, pages 10–18. Springer Verlag, 1985. Lecture Notes in Computer Science, vol. 196.

[89] S. Even and O. Goldreich. On the power of cascade ciphers. *ACM Trans. Comput. Systems*, 3:108–116, 1985.

[90] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signature schemes. In *CRYPTO'89 Proc.* Springer-Verlag, 1990. Lecture Notes in Computer Science, vol. 435.

[91] U. Feige, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988. preliminary version in STOC'87.

[92] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proc. CRYPTO'86*, pages 186–194, Berlin, 1987. Springer Verlag. Lecture Notes in Computer Science, vol. 263.

[93] O. Goldreich. *Foundations of Cryptography (Fragments of a Book)*. Electronic Colloquium on Computational Complexity, 1995. Online access through WWW: http://www.eccc.uni-trier.de/eccc/ or FTP: ftp.eccc.uni-trier.de:/pub/eccc/.

[94] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. In *Proc. of the 25th IEEE Ann. Symp. on Foundations of Computer Science (FOCS)*, pages 464–479, 1984.

[95] O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.

[96] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proc. of the 21st ACM Ann. Symp. on Theory of Computing (STOC)*, pages 25–32, 1989.

[97] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[98] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in $\mathcal{NP}$ have zero-knowledge proof systems. *J. Assoc. Comput. Mach.*, 38(3):691–729, 1991.

[99] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[100] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[101] S. Goldwasser, S. Micali, and P. Tong. Why and how to establish a private code on a public network. In *Proc. of the 23rd IEEE Ann. Symp. on Foundations of Computer Science (FOCS)*, pages 134–144, 1982.

[102] J. Gordon. Strong primes are easy to find. In *EUROCRYPT'84, Lecture Notes in Computer Science, Vol. 209*, pages 216–223, 1985.

[103] J. Hastad. Solving simultaneous modular equations of low degree. *SIAM Journal on Computing*, 17(2):336–341, 1988.

[104] M. E. Hellman. A cryptanalytic time-memory trade off. *IEEE Trans. Inform. Theory*, 26:401–406, 1980.

[105] L. S. Hill. Concerning certain linear transformation apparatus of cryptography. *American Math. Monthly*, 38:135–154, 1931.

[106] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. American Statistical Association*, 58:13–30, 1963.

[107] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudorandom generation from one-way functions. In *Proc. of the 21st ACM Ann. Symp. on Theory of Computing (STOC)*, pages 12–24, 1989.

[108] D. Kahn. *The Codebreakers: The Story of Secret Writing.* Macmillan, New York, 1967.

[109] R. Kannan, A. Lenstra, and L. Lovasz. Polynomial factorization and nonrandomness of bits of algebraic and some transcedental numbers. In *Proc. of the 16th ACM Ann. Symp. on Theory of Computing (STOC)*, pages 191–200, 1984.

[110] J. Köbler. Skript kryptographie: Wintersemester 1993/94. Technical report, Ulm Universität, 1994.

[111] N. Koblitz. *A course in number theory and cryptography*, volume 114 of *Graduate texts in mathematics*. Springer Verlag, New York, 1987. Доступний польський переклад: Wydawnictwa Naukowo-Technichne, Warszawa, 1995.

[112] A. G. Konheim. *Cryptography: A Primer.* John Wiley & Sons, New York, 1981.

[113] R. S. Lehman. Factoring large integers. *Mathematics of Computation*, 28:637–646, 1974.

[114] A. K. Lenstra and Jr. H. W. Lenstra. Algorithms in number theory. In *Handbook of Theoretical Computer Science, Ch. 12*, pages 674–715. Elsevier Publ., 1990.

[115] R. Lipton. How to cheat at mental poker. In *Proc. AMS Short Course on Cryptography.* 1981.

[116] M. Luby and C. Rackoff. Pseudo-random permutation generators and cryptographic composition. In *Proc. of the 18th ACM Ann. Symp. on Theory of Computing (STOC)*, pages 356–363, 1986.

[117] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

[118] M. Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseth, editor, *Advances in cryptology—EUROCRYPT'93 proceedings*, Berlin, 1994. Springer Verlag. Lecture Notes in Computer Science v. 765.

[119] R. C. Merkle. Secure communication over insecure channels. *Communications of the ACM*, 21(4):294–299, 1978.

[120] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proc. of the 21st ACM Ann. Symp. on Theory of Computing (STOC)*, pages 33–43, 1989.

[121] National Bureau of Standards. Announcing the data encryption standard. Tech. Report FIPS Publication 46, 1977.

[122] NIST FIPS PUB XX. Digital signature standard. Technical report, National Institute of Standards and Technology, U.S. Department of Commerce, DRAFT, 1 Feb 1993.

[123] NIST FIPS PUB 180. Secure hash standard. Technical report, National Institute of Standards and Technology, U.S. Department of Commerce, DRAFT, Apr 1993.

[124] T. Okamoto. On relationships between statistical zero-knowledge proofs. In *Proc. of the 28th ACM Ann. Symp. on Theory of Computing (STOC)*, pages 649–658, 1996.

[125] J. Plumstead. Inferring a sequence generated by a linear congruence. In *Proc. of the 23rd IEEE Ann. Symp. on Foundations of Computer Science (FOCS)*, pages 153–159, 1982.

[126] S. Pohlig and M. Hellman. An improved algorithm for computing discrete logarithms over $GF(p)$ and its cryptographic significance. *IEEE Transactions on Information Theory*, 24:106–110, 1978.

[127] C. Pomerance and A. Sárközy. Combinatorial number theory. In *Handbook of Combinatorics*, chapter 20, pages 967–1018. Elsevier Publ., 1995.

[128] C. Pomerance, J. L. Selfridge, and S. S. Wagstaff (Jr.). The pseudoprimes to $25 \cdot 10^9$. *Math. Comput.*, 36(151):1003–1026, 1980.

[129] V. R. Pratt. Every prime has a succinct certificate. *SIAM Journal on Computing*, 4(3):214–220, 1975.

[130] M. O. Rabin. Digitalized signatures and public-key functions as intractible as factorization. Technical Report TR-212, Massachussetts Institute of Technology, Laboratory for Computer Science, 1979.

[131] R. L. Rivest. Cryptography. In *Handbook of Theoretical Computer Science, Ch. 13*, pages 718–755. Elsevier Publ., 1990.

[132] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, 21:120–126, 1978.

[133] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proc. of the 22nd ACM Ann. Symp. on Theory of Computing (STOC)*, pages 387–394, 1990.

[134] B. Rosser. *Proc. London Math. Soc.*, 45(2):21–44, 1939.

[135] J. Rosser and L. Schoenfield. Approximate formulas for some functions of prime numbers. *Illinois J. Math.*, 6:64–94, 1962.

[136] A. Salomaa. *Public-key cryptography.* Springer Verlad, 2 edition, 1996.

[137] B. Schneier. *Applied Cryptography: protocols, algorithms and source code in C.* John Wiley & Sons, New York, 1994. Доступний польський переклад: Wydawnictwa Naukowo-Technichne, Warszawa, 1995.

[138] C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):161–174, 1991. (also in CRYPTO'89 Proc.).

[139] U. Schöning. *Complexity and Structure*, volume 211 of *Lecture Notes in Computer Science.* Springer Verlag, 1985.

[140] A. Shamir. How to share a secret. *Comm. ACM*, 22:612–613, 1979.

[141] G. J. Simmons. *Cryptology*, volume 16, pages 860–873. Encyclopaedia Britannica, Inc., Chicago, 15 edition, 1991.

[142] G. J. Simmons. A "weak" privacy protocol using the RSA cryptosystem. *Cryptologia*, 7(2):180–182, 1983.

[143] A. Sinkov. *Elementary cryptoanalysis — a mathematical approach*, volume 22 of *The New Mathematical Library.* Mathematical Association of America, Washington, 1968.

[144] R. Solovay and V. Strassen. A fast Monte-Carlo test for primality. *SIAM Journal on Computing*, 6(1):84–85, 1977. Erratum, 1978, v. 7, No. 1, p. 118.

[145] J. Spencer. Probabilistic methods. In *Handbook of Combinatorics, Ch. 33*, pages 1785–1817. Elsevier Publ., 1995.

[146] V. Strassen. Algebraic complexity theory. In *Handbook of Theoretical Computer Science, Ch. 11*, pages 633–671. Elsevier Publ., 1990.

[147] A. Tonelli. Bemerkung über die Auflösung quadratischer Congruenzen. *Göttinger Nachrichten*, 344–346, 1891.

[148] U. V. Vazirani and V. V. Vazirani. Efficient and secure pseudo-random number generation. In *Proc. of the 25th IEEE Ann. Symp. on Foundations of Computer Science (FOCS)*, pages 463–485, 1984.

[149] N. K. Vereshchagin. Lower bounds for perceptrons solving some separation problems and oracle separation of AM from PP. Technical Report 498, Computer Science Department, University of Rochester, 1994.

[150] A. Yao. Theory and applications of trapdoor functions. In *Proc. of the 23rd IEEE Ann. Symp. on Foundations of Computer Science (FOCS)*, pages 80–91, 1982.

[151] G. Yuval. How to swindle Rabin. *Cryptologia*, 3(3):187–189, 1979.

Предметний покажчик

$O(f(n))$ – додатнозначна функція $g(n)$ така, що $g(n) \leq cf(n)$ для деякої константи c

$\lceil x \rceil$ – ціле, найближче зверху до дійсного x

$\lfloor x \rfloor$ – ціле, найближче знизу до дійсного x

$\log_k n$ – логарифм за основою k

$\log n$ – логарифм за основою 2

$\ln n$ – натуральний логарифм

$n!$ – факторіал, 20

$a \bmod b$ – остача від ділення a на b, 49

$b \mid a$ – b ділить a, 49

$b \nmid a$ – b не ділить a, 49

НСД (a, b) – найбільший спільний дільник чисел a і b, 50

НСК $(a_1, a_2, \ldots, a_m)$ – найменше спільне кратне чисел $a_1, a_2, \ldots, a_m$, 58

$\mathrm{ind}(x, g, p)$ – дискретний логарифм числа x за основою g і модулем p, 123

$\mathrm{ind}_g x$ – дискретний логарифм числа x за основою g, 123

$\pi(n)$ – кількість простих, не більших від n, 98

$d = (d_l \ldots d_1 d_0)_2$ – запис d у двійковій системі числення

$\oplus$ – операція додавання за модулем 2; 35, 81. Також прямий добуток кілець, 202

$\mathbb{R}$ – множина дійсних чисел

$\mathbb{N}$ – множина натуральних чисел

$\mathbb{Z}$ – множина цілих чисел

$\mathbb{Z}_n$ – кільце лишків за модулем n, 53

$\mathbb{Z}_n^*$ – мультиплікативна група лишків, 53

$\mathbb{J}_n = \mathcal{Q}_n \cup \widetilde{\mathcal{Q}}_n$ – множина лишків x за модулем n з властивістю $\left(\frac{x}{n}\right) = 1$; 100, 140

$\mathcal{Q}_n$ – множина квадратичних лишків за модулем n, 95

$\widetilde{\mathcal{Q}}_n$ – множина псевдоквадратів за модулем n, 100

$\mathcal{A}^*$ – множина слів у алфавіті $\mathcal{A}$, 46

$|w|$ – довжина слова w, 46

$\|X\|$ – кількість елементів у множині X

$x \in X$ – елемент x належить множині X, 198

$X \cap Y$ – перетин множин X і Y, 198

$X \cup Y$ – об'єднання множин X і Y, 198

$X \setminus Y$ – різниця множин X і Y, 198

$X \times Y$ – декартів добуток множин X і Y, 198

id_X – тотожне відображення множини X на себе, 199

D_K – дешифруюче відображення, 14, 47

E_K – шифруюче відображення, 14, 47

$g \circ f$ – композиція відображень f і g, 199

$\det A$ – визначник матриці A, 203

I_k – одинична матриця, 203

$\mathrm{M}_k(R)$ – кільце матриць розміру $k \times k$ над кільцем R, 203

$\mathrm{SL}_k(R)$ – спеціальна лінійна група, 204

$\mathrm{GL}_k(R)$ – повна лінійна група, 203

R^* – мультиплікативна група кільця R, 202

$R[x]$ – кільце многочленів над кільцем R, 205

$\mathrm{Sym}\, X$ – група перестановок множини X, 201

S_n – симетрична група степеня n, 201

$\mathbf{P}[A]$ – ймовірність події A

$\mathbf{E}[Z]$ – математичне сподівання випадкової величини Z

ACM (Association of Computing Machinery), 229
ANSI (American National Standards Institute), 40
ASCII (American Standard Code for Information Interchange), 16
AT&T (American Telephone & Telegraph), 17

BBS генератор (абревіатура від імен авторів: L. Blum, M. Blum, M. Shub), 156–158, 160

DES (Data Encryption Standard), 40–45
DSA (Digital Signature Algorithm), 171–172
DSS (Digital Signature Standard), 148, 171

FOCS (Annual IEEE Symposium on Foundations of Computer Science), 229

IBM (International Business Machines Corporation), 40
IEEE (Institute of Electrical and Electronics Engineers), 229

MAC (message authentication code) – код достовірності, 169
MD5 (від Message Digest), вкорочуюча функція, 170

NBS (US National Bureau of Standards); з 1988 року – NIST (National Institute of Standards and Technology), 40
NIST (US National Institute of Standards and Technology); до 1988 року – NBS (National Bureau of Standards), 171
NSA (US National Security Agency), 170
$\mathcal{NP}$ – клас задач розпізнавання, 187
$\mathcal{NP}$-повна мова ($\mathcal{NP}$-complete language), 189

$\mathcal{P}$ – клас задач розпізнавання, 187

RSA (криптосистема, яку запропонували R. L. Rivest, A. Shamir та L. M. Adleman), 130–137, 151

S-блок (S-box), 42
SHA (від Secure Hash Algorithm), вкорочуюча функція, 170, 171
STOC (Annual ACM Symposium on Theory of Computing), 229

XOR виключне або (від eXclusive OR), 81

Адамар Жак (Hadamard J.), 98
Адлеман Леонард (Adleman L. M.), 130
Алгоритм (algorithm), 75
  детермінований (deterministic), 84
  експоненційний (exponential), 76, 93
  з доступом до оракула (algorithm with access to an oracle), 90
  ймовірнісний (randomized, probabilistic), 84
    із псевдовипадковою послідовністю, 154, 160
    Лас Вегас (Las Vegas), 84, 87, 88
    Монте Карло (Monte Carlo), 84
  недетермінований (nondeterministic), 187
  обчислення, 75
  поліноміальний (polynomial time), 75, 93
  пошуку, 75
  розв'язує задачу, 75, 84
    за час t, 75
  розпізнавання, 75
    відхиляє вхід (rejects), 75
    приймає вхід (accepts), 75
Алгоритм
  бінарного пошуку (binary search), 91
  добування квадратного кореня за простим модулем, 112–113
  Евкліда, 50–51, 58, 205
  Евкліда розширений, 51, 58
  знаходження квадратичного нелишка, 111
  обчислення символу Якобі, 97–99
  піднесення до степеня, 80
  розпізнавання квадратичності за простим модулем, 111
  Сільвера-Поліга-Гелмана, 124–125
  швидкого множення Шонгаге-Штрассена, 82
Алфавіт (alphabet), 46

Ансамблі (ensembles), 153
  непередбачувані (unpredictable), 155
  нерозрізнювані (indistinguishable),153, 159, 194
Атака (attack), 15
  з вибраним відкритим текстом (chosen-plaintext), 15
  з вибраним криптотекстом (chosen-ciphertext), 15
  з відомим відкритим текстом (known-plaintext), 15
  лише із криптотекстом (ciphertext-only), 15
Атака
  брутальна (brute force method), 15
  зустрічна (meet-in-the middle), 39
  словникова (dictionary), 27
Атака проти системи цифрового підпису (attack against digital signatures)
  з вибором повідомлення (chosen-message), 167
  за відомим підписом (known-signature), 167
  за ключем (key-only), 167

Безу Етьен (Bezout E.), 205
Бернуллі Якоб (Bernoulli J.), 209
Бернштейн С. Н., 211
Бертран Ж. Л. Ф. (Bertrand J. L. F.), 98
Біграма (bigram), 22
  нерухома (fixed), 70
Бінарний метод (repeated squaring method), 80
Біт (bit – від binary digit), 35
Блок (block), 26
Блюм Ленора (Blum Lenore), 156
Блюм Мануїл (Blum Manuel), 156, 158
Буква (letter, character), 46
  нерухома (fixed), 68

Важкооборотна функція (one-way function), 146–147
Важкооборотна функція із секретом (trap-door one-way function), 147
  двоїста, 176
Валле Пуссен Ш. Ж. (La Vallee Poussin Ch. J.), 98
Вектор, 203, 206
Векторний простір, 206
Вернам Гілберт (Vernam Gilbert), 17, 35
Визначник, 203
Вимірність лінійного простору, 206
Вихід алгоритму (output), 75
Відкритий текст (plaintext), 12, 47
Відображення
  біективне, 199
  дешифруюче, 47
  ін'єктивне, 198
  обернене, 199
  сюр'єктивне, 198
  тотожне, 199
  шифруюче, 47
  що зберігає довжину (length-preserving), 48
  що зберігає операцію, 200
Віженер, Блез де (Blaise de Vigenère), 17, 23
Вкорочення повідомлення (message digest, fingerprint), 169
Вкорочуюча функція (compression, contraction function), 168
  безколізійна (claw-free), 169
Вхід алгоритму (input), 75
  довжина входу (i. length, i. size), 75

Гасло (password), 148, 185
Гаус Карл Фрідріх (Gauß C. F.), 21, 97, 204
Гелман Мартін (Hellman Martin), 124, 127, 164, 167
Генератор псевдовипадкових послідовностей (pseudo-random sequence generator), 153, див. також *Псевдовипадковий генератор*
Генерування
  ключів (key generation), 128
  простого числа, 107–108, 148
Гілл Лестер С. (Hill Lester S.), 71
Гіпотеза Рімана, 106
  розширена, 106
Гольдрайх Одед (Goldreich Oded), 154
Гольдвассер Шафі (Goldwasser Shafi), 139, 158
Гомоморфізм
  груп, 200
  кілець, 202
Горнер Вільям Джордж (Horner W. G.), 82

Граф, 184, 190
Група, 199
  абелева, 200
  комутативна, 200
  мультиплікативна кільця з одиницею, 202
  перестановок множини, 201
  повна лінійна, 203
  симетрична, 201
  спеціальна лінійна, 204
  циклічна, 200

Двійкове слово (binary word), 35
Дешифрування (decipherment, decryption), 12
Диграф (digraph), 22
Дискретний логарифм, 123
Ділення
  націло, 49
  остача, 49
  частка, 49
Дільник, 49
  найбільший спільний, 50
  одиниці, 202
  тривіальний, 51
Дірихле П. Г. Л. (Dirichlet P. G. L.), 87, 106
Діффі Вайтфілд (Diffie Whitefield), 127, 164, 167
Добуток
  матриць, 203
  шифрів (product of ciphers), 37
Доведення без розголошення (zero-knowledge proof), 180
  в обчислювальному сенсі (computationally), 194, 195
  властивості протоколу
    відсутність розголошення (zero-knowledge), 182–184, 193–194
    обгрунтованість (soundness), 181, 184, 193
    повнота (completeness), 181, 184, 193
  досконале (perfect), 183, 195
  майже досконале (almost perfect), 184
  статистичне (statistical), 184, 195
Дойл Артур Конан (Doyle Arthur Conan), 19

Евклід (Eukléidēs), 50, 108

Елемент
  нейтральний, 199
  обернений, 199
  оборотній (кільця), 202
ЕльГамал Т. (ElGamal T.), 143, 168
Ератосфен (Eratosthénēs), 100
Ефективність (efficiency), 13, 41, 47

Єнсен Йоган Людвіг (Iensen J. L.), 208

Задача (problem)
  $\mathcal{NP}$-повна ($\mathcal{NP}$-complete), 189
  важка, 76
  еквівалентність задач обчислення розпізнавання і пошуку, 74
  індивідуальна, 73, 74
    її розв'язок, 73, 74
  масова, 73
    її звуження, 77
  обчислення, 73
  поліноміально розв'язна (solvable in polynomial time), 75
  пошуку, 73
  розпізнавання, 73
Задача
  дискретного логарифмування (discrete logarithm), 123
  добування квадратного кореня
    за модулем $n = pq$, 113
    за простим модулем, 112
  знаходження квадратичного нелишка за простим модулем, 112
  знаходження первісного кореня за простим модулем, 122
  обчислення функції Ойлера, 118
  пошуку простого поза заданою межею, 109
  розпізнавання квадратичних лишків
    за модулем $n = pq$, 113
    за простим модулем, 111
  розпізнавання первісного кореня за простим модулем, 121
  тестування простоти (primality testing), 100
  факторизації (integer factoring), 109
Звідність задач (reducibility), 90–92
Зведення (reduction), 91

  поліноміальне (polynomial-time reduction), 91
  за Карпом (Karp), 190
  за Куком (Cook), 190
Зведення
  добування квадратного кореня за модулем $n = pq$ до факторизації чисел виду $n = pq$, 114–115
  знаходження первісного кореня за простим модулем до розпізнавання, 122
  обчислення функції Ойлера від $n = pq$ до факторизації $n = pq$, 118
  розпізнавання первісного кореня за простим модулем до факторизації, 121
  факторизації $n = pq$ до обчислення функції Ойлера від $n = pq$, 118
  факторизації чисел виду $n = pq$ до добування квадратного кореня за модулем $n = pq$, 116
  факторизації чисел виду $n = pq$ до знаходження довільного кратного чисел $p - 1$ і $q - 1$, 118–119

Ізоморфізм
  груп, 200
  кілець, 202
Інволюція, 70
Індекс, 123
Ініційний вектор (initialization vector, IV), 43

Ймовірність невдачі Лас Вегас алгоритму, 84
Ймовірність помилки (error probability), 84
  її пониження (reduction), 87
  однобічна (one-sided), 84, 87, 88, 191
  її пониження (reduction), 85

Кардано Джероламо, Іеронімус (Cardano G.), 26, 27
Карп Річард (Karp Richard), 190
Касискі (Kasiski F. W.), 25
Квадратичний закон взаємності, 97
Квадратичний нелишок, 95
Квадратний корінь за модулем n, 95
Кільце, 201
  з одиницею, 202
  комутативне, 202
  лишків за модулем n, 53
Клас $\mathcal{NP}$, 187
Клас $\mathcal{P}$, 187
Клешня вкорочуючої функції (claw), 168
Ключ (key), 13, 47
  відкритий (public), 128
  дешифруючий (deciphering), 60, 127
  таємний (private), 128
  шифруючий (enciphering), 127
Кобліц Ніл (Koblitz Neal), 71
Код (code), 16
Код достовірності (MAC – message authentication code), 169
Коефіцієнт матриці, 203
Колізія вкорочуючої функції (claw), 168
Композиція
  відображень, 199
  шифрів (composition of ciphers), 37, 49
Компонування шифрів (cascading ciphers), 37
Комутативність, 34, 200
Контрольна сума (check sum), 169
Кратне, 49
Криптоаналіз (cryptoanalysis, з грецької – від *kryptós* "сховано" і *analýein* "розв'язувати"), 14
Криптографія (cryptography, з грецької – від *kryptós* "сховано" і *gráphein* "писати"), 14, 16
Криптологія (cryptology, з грецької – від *kryptós* "сховано" і *lógos* "слово"), 15
Криптосистема (cryptosystem), 12, 47
  асиметрична (asymmetric), 129
  з відкритим ключем (public key), 128
  інволютивна, 33, 39
  комутативна (commutative), 177
  симетрична (symmetric), 129
Криптосистема
  RSA, 130–137, 151, 164–166
  Блюма-Гольдвассер, 158–159
  ЕльГамала, 143–145
  Рабіна, 137–139
Криптотекст (ciphertext), 12, 47
Криптування (encryption), 16
  ймовірнісне (probabilistic), 139–140, 158

на основі RSA, 142–143
на основі довільного предикату з секретом, 151
на основі квадратичності, 140–142
кількаразове (multiple), 37
Критерій Ойлера, 96
Кук Стів (Cook Steve), 189, 190

Лагранж Жозеф Луї (Lagrange J. L.), 180, 200, 205
Ламе Габріель (Lamé G.), 59
Левін Леонід, 189
Лежандр А. М. (Legendre A. M.), 95
Леонардо Пізанський, Фібоначчі (Leonardo Pisano, Fibonacci), 58, 83, 101
Лишок, 49
зведений, 53
квадратичний, 95
зведений, 95
Лінійна оболонка, 206
Лінійний генератор (linear congruential generator), 161
Лінійний підпростір, 206
Лінійний простір, 206
Лінійно незалежні вектори, 206

Марков А. А., 209
Матриця, 203
квадратна, 203
невироджена, 207
одинична, 203
унімодулярна, 59
Меркле Ральф (Merkle Ralph), 127
Мерсенн Марен (Mersenne M.), 107, 108
Метод криптоаналізу, див. також Атака
дня народження (birthday), 170, 173
ітерацій, 48, 137
повного перебору (exhausting algorithm), 14, 76
частотний (frequency method), 20–22
Мікелі Сільвіо (Micali Silvio), 139, 154
Міллер Г. Л. (Miller G. L.), 105, 106
Мова, 73
Модуль, 52

Надійність (security), 12
в обчислювальному сенсі (in a computational sense), 17, 41, 140, 189
ймовірнісного криптування, 140, 159
криптосистеми Блюма-Гольдвассер, 159
у теоретико-інформаційному сенсі (in an information-theoretic sense), 36, 45, 140, 180
Надлишковість мови (redundancy), 21, 34
Неоднорідні моделі обчислень (non-uniform computational models), 81, 160
Нерівність
Бернштейна, 211
між середнім геометричним і середнім арифметичним, 208
Маркова, 209
Чебишова, 209
НСД – найбільший спільний дільник, 50
НСК – найменше спільне кратне, 58

Ойлер Леонард (Euler L.), 54, 96, 109
Оракул (oracle), 89
Оцінка Чернова, 210, 221

Паліндром (palindrome), 18, 83
Парадокс із днем народження, 172
Параметр надійності (security parameter), 128, 151
Пароль (password), 148
Паросток (seed), 153
Первісний корінь за модулем p, 94
Період блокового шифру, 26
Перестановка, 26, 41, 42, 201
Підгрупа, 199
По Едгар Алан (Poe E. A.), 34
Повідомлення (message), 12, 47
Подрібнюючі системи (fractionation systems), 37
Покомпонентне виконання операцій, 200
Поліг С. (Pohlig Stephen), 124
Поліграма (polygram), 22
Поліноміально еквівалентні задачі (polynomial time equivalent problems), 92, 100, 151
Поле, 204
Порядок
афінного шифру, 64
групи, 200
елемента групи, 199
квадратної матриці, 203

Послідовність Фібоначчі, 58, 83
Пост Е. Л. (Post E. L.), 74
Постулат Бертрана, 98
Предикат із секретом (trapdoor predicate), 150–151
Префікс, 47
Простір (space)
відкритих текстів (plaintext), 47
ключів (key), 47, 128
криптотекстів (ciphertext), 47
повідомлень (message), 47
Протокол (protocol), 162
доведення без розголошення (zero-knowledge proof)
для квадратичності, 181
для неквадратичності, 183
для розмальовності графа трьома кольорами, 192–194
експоненційного обміну ключем (exponential key exchange), 163–164
заочної гри в карти (mental poker), 177–179
ідентифікації (identification), 185
з використанням гасла, 185
за допомогою симетричної криптосистеми – протокол виклику-і-відгуку (challenge-response protocol), 185
на підставі цифрового підпису, 185
як доведення без розголошення, 185–186
підкидання монети (coin flipping), 174
на основі дволистої важкооборотної функції із секретом, 176
на основі ймовірнісного криптування, 174–176
розподілу секрету (secret sharing), 179–180
цифрового підпису (digital signature), 164
DSA, 171–172
в системі RSA, 164–166
ЕльГамала, 167–168
загальна схема, 166–167
Шнорра, 170–171
Прямий добуток
груп, 200
кілець, 202
Прямолінійна програма (straight-line program), 78
Псевдовипадкові біти (pseudorandom bits), 44
Псевдовипадкова послідовність (pseudorandom sequence), 154
Псевдовипадковий генератор (pseudorandom generator), 153, 154
з розширенням l (with expansion factor l), 153, 159
конструкція на основі важкооборотної перестановки, 155–156
криптографічно надійний (cryptographically secure), 155
непередбачуваність, 155
Псевдоквадрат за модулем n (pseudosquare modulo n), 100, 141

Рабін М. О. (Rabin M. O.), 105, 137, 147
Райвест Рональд (Rivest R. L.), 130, 170
РГР – розширена гіпотеза Рімана, 106
Режими застосування блокових шифрів
електронної кодової книжки (Electronic Code Book – ECB-Mode), 43
зворотнього зв'язку за виходом (Output Feedback – OFB-Mode), 44
зворотнього зв'язку за криптотекстом (Cipher Feedback – CFB-Mode), 44
зчеплення зашифрованих блоків (Cipher Block Chaining – CBC-Mode), 43
простого заміщення (Electronic Code Book – ECB-Mode), 43
Ріман Г. Ф. Б. (Riemann G. F. B.), 89, 106
Розклад на прості співмножники, 52
канонічний, 52
Роторні системи (rotor systems), 38

Символ
Лежандра, 95, 111
Якобі, 96
Сито Ератосфена, 100
Сільвер (Silver), 124
Слово в алфавіті (word over an alphabet), 46
Соловей Р. (Solovay R.), 103
Стірлінг Джеймс (Stirling J.), 20, 208

Сума матриць, 203
Суперник (adversary), 12
Схема (функціональна – circuit), 81, 190
Схема Горнера, 82

Таблиця Віженера (the Vigenère table), 24
Твірний елемент групи, 200
Теорема
  Безу, 205
  Гауса, 204
  Китайська про остачі, 54
  Лагранжа, 200
  Ойлера, 54
  про обернене відображення, 199
  Ферма мала, 54
Тест наступного біту (next-bit test), 155
Тест простоти
  для чисел до $25 \cdot 10^9$, 105
  експоненційний, 101
  квазіполіноміальний, 101
  Лас Вегаса, 104
  Міллера-Рабіна, 105
  поліноміальний за умови РГР, 106
  Соловея-Штрассена, 103
Тонеллі А. (Tonelli A.), 117
Тьюрінг Алан Матісон (Turing A. M.), 17, 74

Фальшування цифрового підпису (forgery of a digital signature)
  вибіркове (selective), 167
  екзистенційне (existential), 167
  повний злам системи підпису (total break), 167
  універсальне (universal), 167
Ферма П'єр (Fermat P.), 54, 108–110
Формула
  для n-го члена послідовності Фібоначчі, 213
  інтерполяційна Лагранжа, 180, 205
  оберненої матриці, 204
  Стірлінга, 208
Функція
  важкооборотна
    EXP, 147, 150, 152
    MULT, 147
    RSA, 147, 150–152, 170
    SQUARE (Рабіна), 147, 150–152, 156, 176, 177
  дзета-функція Рімана, 89, 106
  Ойлера, 54–56

Характеристична функція множини, 74

Цезар Юлій (Julius Caesar), 13
Цикл (перестановка), 201
Цикловий ключ (round key), 41

Час роботи алгоритму (time), 75
  в найгіршому випадку (worst case), 76
  в середньому (expected), 76
  експоненційний (exponential), 76, 191
  ймовірнісного, 84
  поліноміальний (polynomial), 75
Чебишов П. Л., 98, 107, 209
Чернов Г. (Chernoff H.), 210
Числа
  взаємно прості, 50
  конгруентні, 52
  рівні за модулем n, 52
Число
  s-гладке (s-smooth), 124
  Кармайкла (Carmichael), 104, 109
  просте, 51
    Мерсенна, 107, 108
    Ферма, 108, 123
  псевдопросте за основою x, 104, 108, 110
  псевдопросте Ойлера за основою x, 104, 108
  псевдопросте Ферма за основою x, 104
  сильно псевдопросте за основою x, 104, 108, 110
  складене, 51
  ціле Блюма (Blum integer), 117, 139, 147, 151, 159, 177

Шамір Аді (Shamir A.), 130, 179
Шеннон Клод Елвуд (Shannon C. E.), 45
Шифр (cipher), 12, 47
  блоковий (block), 26, 33, 48
  заміни (substitution), 17, 19, 34
    біграмний (bigram), 22
    гомофонний (homophonic), 21
    диграфний (digraph), 22

    поліалфавітний (polyalphabetic), 23
    поліграмний (polygram), 22, 60
    простої (simple), 19, 31, 60
  інволютивний, 33, 39, 70
  підстановки (substitution), 19, див. також *Шифр заміни*
  перестановки (transposition), 17, 19, 26–30, 34
  потоковий (stream), 158
  що утворює групу, 40, 44, 45, 48, 67, 68, 70
Шифр
  ADFGVX, 37
  Enigma, 17, 38
  Playfair, 22
  афінний (affine), 60, 61, 64
  Віженера (Vigenère), 23, 32, 62
  з автоключем (auto-key), 26, 33
  збовтуючий (scrambler), 71
  зсуву (shift), 13, 20, 31, 32, 60
  Кардано, 27, 33, 44
  лінійний (linear), 60, 63
  обходу (route), 26
  обходу матричний, 26, 44
  одноразового блокноту (one-time pad), 17, 35, 44
  Скитала (лат. Scytala), 16, 44
  Цезаря, 13, 32
  частоколу (fence), 13, 44
  чотирьох квадратів, 22, 31
Шифрування (encipherment), 12, див. також *Криптування*
Шнорр Клаус Петер (Schnorr Claus Peter), 170
Шонгаге А. (Schönhage A.), 82
Штрассен Фолькер (Strassen Volker), 82, 103
Шуб Майк (Shub Mike), 156

Ядро важкооборотної функції (hard-core predicate), 149–151, 193
Ядро гомоморфізму, 200, 202
  тривіальне, 200, 202
Якобі К. Г. Я. (Jacobi K. G. J.), 96
Яо А. (Yao A.), 137

Вербіцький Олег Васильович

ВСТУП ДО КРИПТОЛОГІЇ

ВНТЛ — Видавництво науково-технічної літератури

Адреса офісу: 290007, м.Львів, вул. Огієнка 18-а, оф. 1.3.
Адреса для листування: 290007, м. Львів, а/с 1173
тел./факс: (0322) 728073
e-mail: vntl@litech.lviv.ua
http://www.litech.lviv.ua/~vntl

Жовківська друкарня видавництва
Отців Василіян "Місіонер". Зам. 212

292310, м. Жовква Львівської обл.,
вул. Василіянська, 8.