

TAJEMNICA-ATAK-OBRONA



Wydawnictwa  
Naukowo-  
Techniczne

TAO

Douglas R. Stinson

# Kryptografia

W teorii i w praktyce

---

## *Wstęp*

---

Pisałem tę książkę z zamiarem stworzenia wyczerpującego i ogólnego podręcznika, obejmującego wszystkie istotne obszary kryptografii. W ostatnich latach powstało wiele książek i monografii jej poświęconych, jednakże większość z nich dotyczy szczególnych, specjalistycznych zagadnień. Ponadto szybki rozwój badań w dziedzinie kryptografii w ostatnich 15 latach spowodował, że duża część istniejących już podręczników o charakterze ogólnym straciła aktualność.

W University of Nebraska-Lincoln wykładałem kryptografię na poziomie magisterskim dla studentów informatyki, ale zdaje sobie sprawę, że przedmiot ten występuje zarówno na poziomie licencjackim, jak i magisterskim w programach studiów na wydziałach matematycznych, informatycznych i elektrotechniki. Chciałem więc tak zaplanować tę książkę, aby okazała się użyteczna dla Czytelników prezentujących bardzo zróżnicowane podejście do tematu.

Oczywiście próba dotarcia do tak szerokiego audytorium rodzi pewne trudności. W zasadzie starałem się zachować umiar. Wiedzę matematyczną przytoczyłem w rozsądnych granicach tam, gdzie jest ona niezbędna. Wraz z opisami nieformalnymi różnych systemów kryptograficznych podałem także bardziej ścisłe opisy w pseudokodzie, gdyż – jak sądzę – oba podejścia uzupełniają się nawzajem. Zamieściłem też liczne przykłady, by zilustrować działanie algorytmów. I wreszcie, w każdym przypadku podałem próbę wyjaśnienia podtekstu matematycznego, uważam bowiem, że nie można dobrze zrozumieć systemu kryptograficznego, nie rozumiejąc występującej w tle teorii matematycznej.

Książka jest podzielona na trzy części. W pierwszej z nich, rozdziały 1–3, opisałem kryptografię z kluczem prywatnym. W rozdziałach 4–9 przedstawiłem główne zagadnienia kryptografii z kluczem publicznym. W pozostałych czterech rozdziałach zawarłem wprowadzenie do czterech działów kryptografii, w których współczesne badania są szczególnie intensywne.

W pierwszej części książki skupiłem się na następujących tematach: w rozdziale 1 na wprowadzeniu do prostych „klasycznych” systemów kryptograficznych; w rozdziale 2 na zasadniczych elementach podejścia Shannona do kryptografii, włącznie z pojęciem tajności doskonałej i zastosowaniem teorii informacji; a w rozdziale 3 na standardzie szyfrowania danych, DES, oraz kryptoanalizie różnicowej.

Druga część zaczyna się od rozdziału 4, w którym opisałem system kryptograficzny z kluczem publicznym RSA i wprowadziłem wiele tematów z zakresu teorii liczb, takich jak testowanie pierwszości i faktoryzacja liczb naturalnych. W rozdziale 5 przedstawiłem systemy z kluczem publicznym, z których najważniejszy jest system ElGamala oparty na problemie logarytmu dyskretnego. W rozdziale 6 omówiłem schematy podpisów, takie jak standard podpisu cyfrowego, oraz zająłem się szczególnymi rodzajami takich schematów, jak na przykład schematami podpisu niezaprzeczalnego lub niepodrabialnego. Rozdział 7 poświęciłem funkcjom skrótu, a rozdział 8 – przeglądowi wielu różnych metod tworzenia protokołów przydzielania i uzgadniania kluczy. Wreszcie w rozdziale 9 skupiłem się na schematach identyfikacji.

W trzeciej części przedstawiłem wybrane tematy badawcze, w tym dotyczące kodów uwierzytelniania, tajnych schematów współużytkowania, generowania liczb pseudolosowych i dowodów o wiedzy zerowej.

Starałem się obszernie pokazać „jądro” kryptografii, ale również umożliwić Czytelnikowi wejście w kilka bardziej zaawansowanych tematów badawczych. W każdej z omawianych dziedzin próbowałem wybrać kilka reprezentatywnych systemów i przedstawić je z rozsądnią szczegółowością. Tak więc moje podejście do kryptografii nie ma charakteru encyklopedycznego.

Z pewnością książka zawiera znacznie więcej materiału niż potrzeba na jedno- lub nawet dwusemestrальny wykład. Mam jednak nadzieję, że może ona posłużyć za podstawę do kilku różnych wykładów. I tak, wykład podstawowy mógłby ograniczać się do treści rozdziału 1 wraz z wybranymi tematami z rozdziałów 2–5. Pełniej zagadnienia te mogłyby zostać przedstawione na wykładzie dla zaawansowanych lub na poziomie magisterskim. Podobnie rzecz się ma z materiałem z rozdziałów 6–9. Ponadto każdy z rozdziałów mógłby stanowić podstawę do wykładu monograficznego, wchodzącego już głębiej w określony obszar kryptografii.

Liczę jednakże i na to, że oprócz pełnienia funkcji podręcznika niniejsza książka będzie użyteczna także dla teoretyków i praktyków kryptografii, jako wprowadzenie do tych jej zakątek, których jeszcze dobrze nie poznali. Z tą myślą zamieściłem obszerną bibliografię, będącą rozwinięciem wielu poruszonych tu tematów.

Jednym z najtrudniejszych problemów, stojących przed autorem książki takiej jak ta, jest określenie zakresu matematyki, którą należy Czytelnikowi przedstawić. Kryptografia jest dziedziną obszerną i wymagającą znajomości różnych gałęzi matematyki, w tym teorii liczb, teorii grup, pierścieni i ciał, algebry liniowej, teorii prawdopodobieństwa oraz teorii informacji. Przydatna jest również znajomość innych dziedzin, takich jak złożoność obliczeniowa, algorytmika i teoria NP-zupełności. Starałem się nie zakładać, że Czytelnik jest w tych sprawach ekspertem i w związku z tym wprowadzałem w miarę potrzeby odpowiednie narzędzia matematyczne. Jednakże pewna wiedza w zakresie podstaw algebry liniowej i arytmetyki modularnej bez wątpienia Czytelnikowi się przyda. Tematy bardziej specjalistyczne, jak na przykład pojęcie entropii w teorii informacji, wprowadziłem od podstaw.

Powiniensem również przeprosić tych, którzy nie zgadzają się z określeniem „w teorii i w praktyce” zawartym w tytule. Przyznaję, że jest tu więcej teorii niż praktyki. Tytuł ma jednak wyrażać mój zamiar zawarcia w książce tematów zarówno interesujących z teoretycznego punktu widzenia, jak i istotnych dla praktyki. Mogłem więc umieścić tu opisy systemów niezbyt praktycznych, ale eleganckich matematycznie lub dobrze ilustrujących pewne pojęcia bądź metody, a także omówić najważniejsze systemy stosowane w praktyce, na przykład DES oraz inne standardy kryptograficzne stosowane w USA.

Chciałbym tu podziękować wielu osobom, które podtrzymywały mnie na duchu w trakcie pracy nad książką, wytykały błędy lub pomyłki, wskazywały tematy, które warto przedstawić, lub sugerowały, jak je potraktować. W szczególności na wyrazy wdzięczności zasługują Mustafa Atici, Mihir Bellare, Bob Blakley, Carlo Blundo, Gilles Brassard, Daniel Ducharme, Mike Dvorsky, Luiz Frota-Mattos, David Klarner, Don Kreher, Keith Martin, Vaclav Matyas, Alfred Menezes, Luke O’Connor, William Read, Phil Rogaway, Paul Van Oorschot, Scott Vanstone, Johan van Tilburg, Marc Vauclair oraz Mike Wiener. Dziękuję również Mike’owi Dvorsky’emu za pomoc w przygotowaniu skorowidza.

*Douglas R. Stinson*

---

# *Spis treści*

---

<b>1. Klasyczna kryptografia</b>	<b>1</b>
1.1. Wprowadzenie: kilka prostych kryptosystemów . . . . .	1
1.1.1. Szyfr z przesunięciem . . . . .	3
1.1.2. Szyfr podstawieniowy . . . . .	7
1.1.3. Szyfr afiniczny . . . . .	8
1.1.4. Szyfr Vigenèrè'a . . . . .	12
1.1.5. Szyfr Hilla . . . . .	14
1.1.6. Szyfr permutujący . . . . .	18
1.1.7. Szyfry strumieniowe . . . . .	20
1.2. Kryptoanaliza . . . . .	25
1.2.1. Kryptoanaliza szyfru afanicznego . . . . .	27
1.2.2. Kryptoanaliza szyfru podstawieniowego . . . . .	28
1.2.3. Kryptoanaliza szyfru Vigenèrè'a . . . . .	31
1.2.4. Atak ze znanym tekstem jawnym na szyfr Hilla . . . . .	37
1.2.5. Kryptoanaliza szyfru strumieniowego opartego na LFSR . . . . .	38
1.3. Uwagi . . . . .	40
Ćwiczenia . . . . .	40
<b>2. Teoria Shannona</b>	<b>45</b>
2.1. Tajność doskonała . . . . .	45
2.2. Entropia . . . . .	52
2.2.1. Kodowania Huffmana a entropia . . . . .	54
2.3. Własności entropii . . . . .	57
2.4. Błędne klucze i długość krytyczna . . . . .	60
2.5. Produktowe systemy kryptograficzne . . . . .	65
2.6. Uwagi . . . . .	68
Ćwiczenia . . . . .	69
<b>3. Standard szyfrowania danych - DES</b>	<b>72</b>
3.1. Wprowadzenie . . . . .	72
3.2. Opis DES . . . . .	72
3.2.1. Przykład szyfrowania w systemie DES . . . . .	81

3.3.	Kontrowersje wokół DES . . . . .	83
3.4.	DES w praktyce . . . . .	85
3.4.1.	Tryby działania DES . . . . .	86
3.5.	Wymagania czasowe a wymagania pamięciowe – próba kompromisu . . . . .	89
3.6.	Kryptoanaliza różnicowa . . . . .	91
3.6.1.	Atak na 3-rundowy szyfr DES . . . . .	95
3.6.2.	Atak na 6-rundowy szyfr DES . . . . .	100
3.6.3.	Inne przykłady kryptoanalizy różnicowej . . . . .	111
3.7.	Uwagi i bibliografia . . . . .	112
	Ćwiczenia . . . . .	112
<b>4.</b>	<b>System RSA i faktoryzacja</b>	<b>116</b>
4.1.	Wprowadzenie do kryptografii z kluczem publicznym . . . . .	116
4.2.	Jeszcze o teorii liczb . . . . .	118
4.2.1.	Algorytm Euklidesa . . . . .	118
4.2.2.	Chińskie twierdzenie o resztach . . . . .	121
4.2.3.	Dalsze pożyteczne fakty . . . . .	124
4.3.	System kryptograficzny RSA . . . . .	125
4.4.	Implementacja RSA . . . . .	127
4.5.	Probabilistyczny test na liczby pierwsze . . . . .	130
4.6.	Ataki na RSA . . . . .	139
4.6.1.	Wykładnik deszyfrowania . . . . .	140
4.6.2.	Częściowa informacja o bitach tekstu jawnego . . . . .	145
4.7.	System kryptograficzny Rabina . . . . .	147
4.8.	Algorytmy faktoryzacji . . . . .	152
4.8.1.	Metoda $p - 1$ . . . . .	152
4.8.2.	Algorytm Dixona i sito kwadratowe . . . . .	154
4.8.3.	Algorytmy faktoryzacji w praktyce . . . . .	156
4.9.	Uwagi i bibliografia . . . . .	157
	Ćwiczenia . . . . .	158
<b>5.</b>	<b>Inne systemy kryptograficzne z kluczem publicznym</b>	<b>164</b>
5.1.	System kryptograficzny ElGamala i logarytmy dyskretnie . . . . .	164
5.1.1.	Algorytmy dla problemu logarytmu dyskretnego . . . . .	166
5.1.2.	Bezpieczeństwo bitów logarytmu dyskretnego . . . . .	174
5.2.	Systemy oparte na ciałach skończonych i krzywych eliptycznych . . . . .	179
5.2.1.	Ciąła Galois . . . . .	182
5.2.2.	Krzywe eliptyczne . . . . .	185
5.3.	System plecakowy Merkle'a-Hellmana . . . . .	192
5.4.	System McEliece'a . . . . .	195
5.5.	Uwagi i bibliografia . . . . .	200
	Ćwiczenia . . . . .	201

<b>6. Schematy podpisów</b>	<b>204</b>
6.1. Wprowadzenie . . . . .	204
6.2. Schemat podpisu ElGamala . . . . .	207
6.3. Standard podpisu cyfrowego . . . . .	212
6.4. Jednorazowe podpisy . . . . .	215
6.5. Podpisy niezaprzeczalne . . . . .	220
6.6. Podpisy niepodrabialne . . . . .	226
6.7. Uwagi i bibliografia . . . . .	231
Ćwiczenia . . . . .	232
<b>7. Funkcje skrótu</b>	<b>235</b>
7.1. Podpisy i funkcje skrótu . . . . .	235
7.2. Bezkonfliktowe funkcje skrótu . . . . .	236
7.3. Atak metodą dnia urodzin . . . . .	239
7.4. Funkcja skrótu związana z logarytmem dyskretnym . . . . .	241
7.5. Rozszerzone funkcje skrótu . . . . .	244
7.6. Funkcje skrótu budowane na systemach kryptograficznych . . . . .	249
7.7. Funkcja skrótu MD4 . . . . .	250
7.8. Datowanie . . . . .	255
7.9. Uwagi i bibliografia . . . . .	257
Ćwiczenia . . . . .	258
<b>8. Uzgadnianie i dystrybucja klucza</b>	<b>260</b>
8.1. Wprowadzenie . . . . .	260
8.2. Wstępna dystrybucja klucza . . . . .	262
8.2.1. Schemat Bloma . . . . .	262
8.2.2. Schemat wstępnej dystrybucji klucza Diffiego–Hellmana . . . . .	265
8.3. Kerberos . . . . .	269
8.4. Protokół wymiany kluczy Diffiego–Hellmana . . . . .	271
8.4.1. Protokół wymiany klucza z uwierzytelnieniem . . . . .	272
8.4.2. Protokół uzgadniania klucza MTI . . . . .	275
8.4.3. Uzgadnianie klucza z użyciem klucza samopotwierdzającego . . . . .	278
8.5. Uwagi i bibliografia . . . . .	281
Ćwiczenia . . . . .	282
<b>9. Schematy identyfikacji</b>	<b>284</b>
9.1. Wprowadzenie . . . . .	284
9.2. Schemat identyfikacji Schnorra . . . . .	286
9.3. Schemat identyfikacji Okamoto . . . . .	291
9.4. Schemat identyfikacji Guillou–Quisquatera . . . . .	297
9.4.1. Schematy identyfikacji oparte na tożsamości . . . . .	301
9.5. Przekształcenie identyfikacji w schemat podpisu . . . . .	302
9.6. Uwagi i bibliografia . . . . .	303
Ćwiczenia . . . . .	303

<b>10. Kody uwierzytelniania</b>	<b>305</b>
10.1. Wprowadzenie . . . . .	305
10.2. Obliczanie prawdopodobieństw oszustwa . . . . .	307
10.3. Ograniczenia kombinatoryczne . . . . .	312
10.3.1. Macierze ortogonalne . . . . .	314
10.3.2. Macierze ortogonalne – konstrukcje i ograniczenia . . . . .	315
10.3.3. Charakteryzacje kodów uwierzytelniania . . . . .	320
10.4. Ograniczenia związane z entropią . . . . .	321
10.5. Uwagi i bibliografia . . . . .	324
Ćwiczenia . . . . .	324
<b>11. Tajne schematy współużytkowania</b>	<b>326</b>
11.1. Wprowadzenie: schemat progowy Shamira . . . . .	326
11.2. Struktury dostępu i ogólne tajne współużytkowanie . . . . .	331
11.3. Konstrukcja oparta na obwodzie monotonicznym . . . . .	333
11.4. Formalne definicje . . . . .	338
11.5. Względna miara informacji . . . . .	342
11.6. Konstrukcja Brickella oparta na przestrzeni liniowej . . . . .	343
11.7. Górne ograniczenie względnej miary informacji . . . . .	349
11.8. Konstrukcja przez rozkład . . . . .	353
11.9. Uwagi i bibliografia . . . . .	357
Ćwiczenia . . . . .	358
<b>12. Generowanie liczb pseudolosowych</b>	<b>360</b>
12.1. Wprowadzenie i przykłady . . . . .	360
12.2. Nierozróżnialne rozkłady prawdopodobieństwa . . . . .	364
12.2.1. Algorytm przewidywania następnego bitu . . . . .	367
12.3. Generator Bluma–Bluma–Shuba . . . . .	371
12.3.1. Bezpieczeństwo generatora BBS . . . . .	373
12.4. Szyfrowanie probabilistyczne . . . . .	379
12.5. Uwagi i bibliografia . . . . .	384
Ćwiczenia . . . . .	384
<b>13. Dowody o wiedzy zerowej</b>	<b>387</b>
13.1. Interakcyjne systemy dowodowe . . . . .	387
13.2. Doskonale dowody o wiedzy zerowej . . . . .	390
13.3. Schemat wiążącego bitu . . . . .	400
13.4. Dowody o obliczeniowej wiedzy zerowej . . . . .	402
13.5. Rozumowanie o wiedzy zerowej . . . . .	407
13.6. Uwagi i bibliografia . . . . .	409
Ćwiczenia . . . . .	409
<b>Dalsze lektury</b>	<b>411</b>
<b>Bibliografia</b>	<b>413</b>
<b>Skorowidz</b>	<b>426</b>

# 1

---

## Klasyczna kryptografia

---

### 1.1. Wprowadzenie: kilka prostych kryptosystemów

Podstawowy cel, jaki stawia sobie kryptografia, to umożliwienie dwóm osobom, które będziemy odtąd nazywać Alicją i Bolkiem, porozumiewania się za pośrednictwem niechronionego kanału, tak aby przekazywane treści pozostały niezrozumiałe dla ich przeciwnika Oskara. Kanałem może być, na przykład, linia telefoniczna lub sieć komputerowa, a informacją, którą Alicja chce przekazać Bolkowi, zwaną dalej *tekstem jawnym*, może być tekst słowny w dowolnym języku, dane liczbowe lub cokolwiek innego – jej struktura nie ma tu żadnego znaczenia. Alicja szyfruje tekst jawnego, używając z góry ustalonego klucza, po czym przesyła otrzymany w ten sposób *tekst zaszyfrowany* (czyli *kryptogram*) poprzez wybrany kanał. Oskar, przechwytyując ten zaszyfrowany tekst w kanale, nie jest w stanie odtworzyć źródłowego tekstu jawnego. Może to natomiast zrobić Bolek, który zna klucz szyfru.

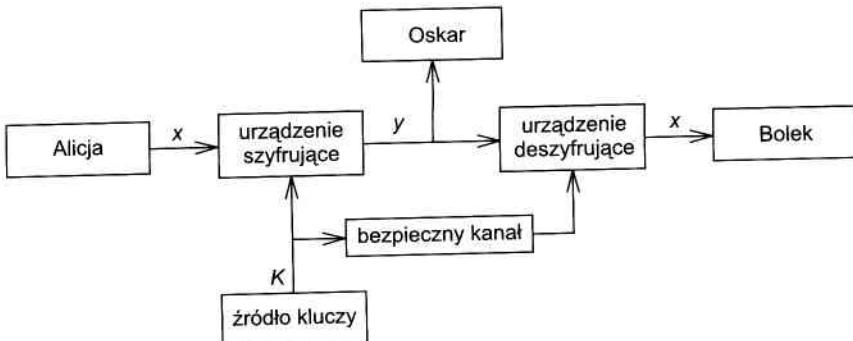
Tę sytuację opiszymy ściślej za pomocą zapisu matematycznego.

#### DEFINICJA 1.1

Systemem kryptograficznym nazywamy dowolną piątkę  $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ , dla której spełnione są następujące warunki:

- (1)  $\mathcal{P}$  jest skończonym zbiorem możliwych tekstów jawnych.
- (2)  $\mathcal{C}$  jest skończonym zbiorem możliwych tekstów zaszyfrowanych.
- (3)  $\mathcal{K}$ , przestrzeń kluczy, jest skończonym zbiorem możliwych kluczy.
- (4) Dla każdego  $K \in \mathcal{K}$  istnieje reguła szyfrowania  $e_K \in \mathcal{E}$  i odpowiadająca jej reguła deszyfrowania  $d_K \in \mathcal{D}$ . Reguły  $e_k : \mathcal{P} \rightarrow \mathcal{C}$  i  $d_k : \mathcal{C} \rightarrow \mathcal{P}$  są funkcjami, takimi że  $d_k(e_k(x)) = x$  dla każdego tekstu jawnego  $x$ .

Najistotniejszy jest tu warunek 4. Stwierdza on, że jeśli do szyfrowania tekstu jawnego  $x$  użyto funkcji  $e_K$ , a uzyskany kryptogram deszyfrowano za pomocą funkcji  $d_K$ , to wynikiem tego procesu będzie pierwotny tekst jawnny  $x$ .



RYSUNEK 1.1. Kanał komunikacyjny

Alicja i Bolek, aby użyć pewnego ustalonego systemu kryptograficznego, stosują następującą procedurę. Najpierw wybiorą losowo klucz  $K \in \mathcal{K}$ . Zrobią to wtedy, gdy będą przebywać jednocześnie w tym samym miejscu, gdzie Oskar nie może ich śledzić, albo gdy uzyskają dostęp do bezpiecznego kanału komunikacyjnego (w tym przypadku mogą znajdować się w różnych miejscach). Przypuśćmy, że w późniejszym czasie Alicia chce przekazać Bolekowi wiadomość za pośrednictwem niepewnego kanału i że wiadomość ta ma postać ciągu znaków:

$$\mathbf{x} = x_1 x_2 \dots x_n$$

dla pewnej liczby naturalnej  $n \geq 1$ , gdzie każdy symbol  $x_i$  tekstu jawnego należy do  $\mathcal{P}$  i jest utajniony za pomocą reguły szyfrowania  $e_K$ , wyznaczonej przez ustalony klucz  $K$ . Tak więc Alicia oblicza  $y_i = e_K(x_i)$ ,  $1 \leq i \leq n$ , i uzyskany ciąg tekstu zaszyfrowanego

$$\mathbf{y} = y_1 y_2 \dots y_n$$

przesyła kanałem komunikacyjnym. Gdy do Bolka dotrze tekst  $y_1 y_2 \dots y_n$ , ten odtajni go za pomocą funkcji deszyfrującej  $d_K$ , otrzymując pierwotny ciąg tekstu jawnego, czyli  $x_1 x_2 \dots x_n$ . Rysunek 1.1 ilustruje schemat kanału komunikacyjnego.

Oczywiście, funkcja szyfrująca  $e_K$  musi być różnowartościowa, gdyż w przeciwnym przypadku jednoznaczne odczytanie tekstu zaszyfrowanego nie byłoby możliwe. Na przykład, gdy

$$y = e_K(x_1) = e_K(x_2),$$

gdzie  $x_1 \neq x_2$ , Bolek nie jest w stanie stwierdzić, czy  $y$  jest szyfrogramem odpowiadającym  $x_1$  czy  $x_2$ . Zauważmy, że jeśli  $\mathcal{P} = \mathcal{C}$ , to każda funkcja szyfrująca jest permutacją. Inaczej mówiąc, jeśli zbiory tekstu jawnego i tekstu zaszyfrowanego są równe, to każda funkcja szyfrująca po prostu zmienia uporządkowanie (dokonuje permutacji) elementów tego zbioru.

### 1.1.1. Szyfr z przesunięciem

W tym punkcie opiszymy **szyfr z przesunięciem** (albo krócej, **przesunięcie**) oparty na arytmetyce reszt. Przypomnijmy najpierw kilka podstawowych pojęć tej arytmetyki.

#### DEFINICJA 1.2

Niech  $a$  i  $b$  będą liczbami całkowitymi, a  $m$  liczbą całkowitą dodatnią. Mówimy, że „liczba  $a$  przystaje do  $b$  modulo  $m$ ” i piszemy  $a \equiv b \pmod{m}$ , gdy  $m$  jest dzielnikiem liczby  $b - a$ . Liczbę  $m$  nazywamy *modułem*.

Kiedy dzielimy liczby  $a$  i  $b$  przez  $m$ , otrzymujemy całkowity iloraz oraz całkowitą resztę zawartą między  $0$  i  $m - 1$ . Inaczej mówiąc,  $a = q_1m + r_1$  i  $b = q_2m + r_2$ , gdzie  $0 \leq r_1 \leq m - 1$  i  $0 \leq r_2 \leq m - 1$ . Nietrudno zauważyc, że  $a \equiv b \pmod{m}$  wtedy i tylko wtedy, gdy  $r_1 = r_2$ . Resztę z dzielenia  $a$  przez  $m$ , czyli wartość  $r_1$  z powyższych równań, będziemy oznaczać symbolem  $a \bmod m$  (bez nawiasów). Tak więc  $a \equiv b \pmod{m}$  wtedy i tylko wtedy, gdy  $a \bmod m = b \bmod m$ . Zastępując  $a$  przez  $a \bmod m$ , powiemy, że liczba  $a$  została *zredukowana modulo  $m$* .

**UWAGA** Wiele języków programowania definiuje  $a \bmod m$  jako resztę należącą do przedziału od  $-m + 1$  do  $m - 1$  i o tym samym znaku co  $a$ . Na przykład,  $-18 \bmod 7$  jest wtedy równe  $-4$  zamiast  $3$ , jak wynika z przyjętej wyżej definicji. Jednak dla naszych celów znacznie wygodniej będzie założyć, że reszta jest zawsze nieujemna. ■

Możemy teraz zdefiniować arytmetykę reszt modulo  $m$ . Niech  $\mathbb{Z}_m$  oznacza zbiór  $\{0, \dots, m - 1\}$  z dwoma działaniami:  $+$  i  $\cdot$ . Dodawanie i mnożenie jest określone w  $\mathbb{Z}_m$  dokładnie tak, jak w zbiorze liczb rzeczywistych – z tą różnicą, że wyniki redukujemy modulo  $m$ .

Na przykład, przyjmijmy, że chcemy obliczyć  $11 \cdot 13$  w  $\mathbb{Z}_{16}$ . Przy zwykłym mnożeniu mamy  $11 \cdot 13 = 143$ . Aby teraz uzyskać redukcję tej liczby modulo 16, musimy wykonać dzielenie z resztą:  $143 = 8 \cdot 16 + 15$ . Tak więc  $143 \bmod 16 = 15$ , a stąd  $11 \cdot 13 = 15$  w  $\mathbb{Z}_{16}$ .

Tak określone działania dodawania i mnożenia w  $\mathbb{Z}_m$  spełniają większość dobrze znanych praw arytmetyki. Wymienimy je tu bez dowodu:

- (1) dodawanie *nie wyprowadza* poza zbiór  $\mathbb{Z}_m$ , czyli  $a + b \in \mathbb{Z}_m$ , gdy  $a, b \in \mathbb{Z}_m$ ;
- (2) dodawanie jest *przemienne*, czyli dla dowolnych  $a, b \in \mathbb{Z}_m$ ,  $a + b = b + a$ ;
- (3) dodawanie jest *łączne*, czyli dla dowolnych  $a, b, c \in \mathbb{Z}_m$ ,  $(a + b) + c = a + (b + c)$ ;
- (4)  $0$  jest elementem *neutralnym* dodawania, czyli dla dowolnego  $a \in \mathbb{Z}_m$ ,  $a + 0 = 0 + a = a$ ;
- (5) elementem *przeciwnym* dowolnego  $a \in \mathbb{Z}_m$  jest  $m - a$ , czyli  $a + (m - a) = (m - a) + a = 0$  dla każdego  $a \in \mathbb{Z}_m$ ;

- (6) mnożenie *nie wyprowadza* poza zbiór  $\mathbb{Z}_m$ , czyli  $ab \in \mathbb{Z}_m$ , gdy  $a, b \in \mathbb{Z}_m$ ;
- (7) mnożenie jest *przemienne*, czyli dla dowolnych  $a, b \in \mathbb{Z}_m$ ,  $ab = ba$ ;
- (8) mnożenie jest *łaczne*, czyli dla dowolnych  $a, b, c \in \mathbb{Z}_m$ ,  $(ab)c = a(bc)$ ;
- (9) 1 jest elementem *neutralnym* mnożenia, czyli dla dowolnego  $a \in \mathbb{Z}_m$ ,  $a \cdot 1 = 1 \cdot a = a$ ;
- (10) mnożenie jest *rozdzielne* względem dodawania, czyli dla dowolnych  $a, b, c \in \mathbb{Z}_m$ ,  $(a + b)c = (ac) + (bc)$  oraz  $a(b + c) = (ab) + (ac)$ .

Właściwości 1, 3–5 świadczą o tym, że  $\mathbb{Z}_m$  stanowi strukturę algebraiczną zwaną *grupą* ze względu na operację dodawania. Ponieważ grupa ta ma także właściwość 2, mówimy, że jest to grupa *abelowa*.

W istocie właściwości 1–10 stwierdzają, że  $\mathbb{Z}_m$  jest *pierścieniem*. W książce zobaczymy wiele dalszych przykładów grup i pierścieni. Pierścieniami są na przykład takie znajome struktury jak zbiór liczb całkowitych  $\mathbb{Z}$ , zbiór liczb rzeczywistych  $\mathbb{R}$  oraz zbiór liczb zespolonych  $\mathbb{C}^{\dagger}$ . Są to jednak pierścienie nieskończone, podczas gdy my skupimy się niemal wyłącznie na pierścieniach skończonych.

Dzięki istnieniu dla każdego elementu  $\mathbb{Z}_m$  elementu doń przeciwnego możemy w tym zbiorze wykonywać także odejmowanie. Różnicę  $a - b$  w  $\mathbb{Z}_m$  definiujemy jako element  $a + m - b \bmod m$ . Równoważnie, możemy ją określić jako liczbę  $a - b$  zredukowaną modulo  $m$ .

Na przykład, w celu uzyskania różnicy  $11 - 18$  w  $\mathbb{Z}_{31}$  obliczamy  $11 + 31 \bmod 31 = 24$ . Możemy również najpierw odjąć 18 od 11, a następnie zredukować  $-7$  modulo 31:  $-7 \bmod 31 = 24$ .

Szyfr z przesunięciem przedstawiamy na rysunku 1.2. Jest on określony na  $\mathbb{Z}_{26}$ , ponieważ angielski alfabet składa się z 26 liter<sup>†</sup>, choć równie dobrze moglibyśmy użyć pierścienia  $\mathbb{Z}_m$  dla dowolnego modułu  $m$ . Łatwo zauważyc, że

Niech  $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}$ . Dla  $0 \leq K \leq 25$  definiujemy

$$e_K(x) = x + K \bmod 26$$

oraz

$$d_K(y) = y - K \bmod 26$$

$$(x, y \in \mathbb{Z}_{26}).$$

### RYSUNEK 1.2. Szyfr z przesunięciem

<sup>†</sup>W każdym przypadku z działaniami dodawania i mnożenia w danym zbiorze (przyp. tłum.).

<sup>‡</sup>Ze względu na istotną z kryptograficznego punktu widzenia specyfikę języka angielskiego (częstości występowania liter itp.) w całej książce zachowano przykłady i odniesienia do tego języka (przyp. tłum.).

przesunięcie jest systemem kryptograficznym w sensie podanej wcześniej definicji: czyli  $d_K(e_K(x)) = x$  dla każdego  $x \in \mathbb{Z}_{26}$ .

**UWAGA** Gdy  $K = 3$ , system kryptograficzny często jest nazywany **szyfrem Cezara**, gdyż posługiwał się nim ponoć Juliusz Cezar. ■

Przystępując do utajniania zwykłego angielskiego tekstu za pomocą przesunięcia (z modułem 26), ustalimy odpowiedniość między znakami alfabetu a resztami modulo 26 w następujący sposób:  $A \leftrightarrow 0, B \leftrightarrow 1, \dots, Z \leftrightarrow 25$ . Będziemy korzystać z tej odpowiedniości w wielu przykładach, więc odnotujmy ją tu do przyszłego użytku:

$A$	$B$	$C$	$D$	$E$	$F$	$G$	$H$	$I$	$J$	$K$	$L$	$M$
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

Prosty przykład ilustruje sposób wykorzystania tej odpowiedniości.

### Przykład 1.1

Przyjmijmy, że kluczem szyfru z przesunięciem jest  $K = 11$ , a tekstem jawnym ciąg

wewillmeetatmidnight<sup>†</sup>.

Najpierw przekształcamy ten tekst w ciąg liczb, zgodnie z opisaną wyżej odpowiedniością:

22	4	22	8	11	11	12	4	4	19
0	19	12	8	3	13	8	6	7	19

Następnie dodajemy 11 do każdej wartości i redukujemy otrzymane sumy modulo 26:

7	15	7	19	22	22	23	15	15	4
11	4	23	19	14	24	19	17	18	4

Teraz przekształcamy ten ciąg liczb w znaki alfabetu i mamy tekst zaszyfrowany:

HPHTWWXPPELEXTOYTRSE.

Aby odczytać ten tekst, Bolek przekształci go najpierw w ciąg liczb, potem od każdej z nich odejmie 11 (redukując wynik modulo 26), a na koniec otrzymany ciąg liczb przekształci ponownie do postaci ciągu znaków alfabetu. ■

---

<sup>†</sup>W tłumaczeniu „spotkamy się o północy” (przyp. tłum.).

**UWAGA** W przykładzie użyliśmy małych liter w tekście jawnym i wielkich w tekście zaszyfrowanym. W dalszych przykładach będziemy czynić podobnie w celu polepszenia czytelności. ■

Jeśli system kryptograficzny ma być użyteczny, powinien spełniać kilka warunków. Dwa z nich przedstawimy nieformalnie teraz.

- (1) Każda z funkcji szyfrujących  $e_K$  oraz każda z funkcji deszyfrujących  $d_K$  powinna być efektywnie obliczalna.
- (2) Zarówno klucz  $K$  użyty do przesłania wiadomości, jak i tekst jawnny  $\mathbf{x}$  nie mogą być odkryte przez przeciwnika, nawet gdy przechwyci on tekst zaszyfrowany  $\mathbf{y}$ .

Druga z tych własności określa w bardzo mglisty sposób pojęcie „bezpieczeństwa”. Proces prób zmierzających do obliczenia klucza  $K$  na podstawie danego tekstu zaszyfrowanego  $\mathbf{y}$  nazywa się *kryptoanalizą*. (Dalej te pojęcia opiszymy dokładniej). Rzecz jasna, gdyby Oskar był w stanie ustalić klucz  $K$ , potrafiłby odczytać tekst zaszyfrowany  $\mathbf{y}$  nie gorzej od Bolka, korzystając z funkcji  $d_K$ . Stąd ustalenie  $K$  powinno być co najmniej równie tudne, jak ustalenie tekstu jawnego  $\mathbf{x}$ .

Zauważmy, że przesunięcie (modulo 26) nie jest bezpieczne, ponieważ można je poddać kryptoanalizie oczywistą metodą *wyczerpującego przeszukiwania kluczy*. Istnieje tylko 26 możliwych kluczy, zatem bez trudu można wypróbować każdą z odpowiadających im możliwych reguł deszyfrowania  $d_K$ , dopóki nie otrzyma się „sensownego” tekstu jawnego. Ilustruje to następny przykład.

### Przykład 1.2

Do tekstu zaszyfrowanego

JBCRCLQRWRVNBJENBWRWN

zastosujemy kolejno klucze  $d_0, d_1$  itd. Otrzymamy następujące ciągi:

```
jbcrclqrwcrvnbjenbwrvn
iabqbkpqvqbqumaidmavqvṁ
hzapajopuaptlzhclzupul
gyzozinotzoskygbkytotk
fxynyhmnssynrjxfajxsnsj
ewxmxglmrxmlqiweziwrmi
dvwlwfklqlphvdyhvqlqh
cuvkvejkpvkogucxgupkpg
btujudijoujnftbwftojof
astitchintimesavesnine†
```

W tym miejscu ustaliliśmy tekst jawny i możemy się zatrzymać. Kluczem jest  $K = 9$ . ■

---

†W wolnym tłumaczeniu: „Zaszyj dziurkę póki mała” (przyp. tum.).

Tekst jawnny zostanie obliczony średnio po wypróbowaniu  $26/2 = 13$  reguł deszyfrowania.

Jak pokazuje powyższy przykład, koniecznym warunkiem bezpieczeństwa systemu kryptograficznego jest niewykonalność metody wyczerpującego przeszukiwania; inaczej mówiąc, przestrzeń kluczowa powinna być bardzo duża. Jak można się spodziewać, wielkość przestrzeni kluczowej nie jest jednak warunkiem wystarczającym do zapewnienia bezpieczeństwa.

### 1.1.2. Szyfr podstawieniowy

Kolejnym dobrze znanym systemem kryptograficznym jest używany od stuleci **szyfr podstawieniowy**. Przykładami zastosowań tego szyfru są pojawiające się w czasopismach kryptogramy-łamigłówki. Opis szyfru podstawieniowego znajduje się na rysunku 1.3.

Niech  $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}$ .  $\mathcal{K}$  składa się ze wszystkich permutacji 26 symboli  $0, 1, \dots, 25$ . Dla każdej permutacji  $\pi \in \mathcal{K}$  definiujemy

$$e_\pi(x) = \pi(x)$$

oraz

$$d_\pi(y) = \pi^{-1}(y),$$

gdzie  $\pi^{-1}$  jest permutacją odwrotną do  $\pi$ .

**RYSUNEK 1.3. Szyfr podstawieniowy**

W przypadku szyfru podstawieniowego możemy przyjąć, że zarówno zbiór  $\mathcal{P}$ , jak i zbiór  $\mathcal{C}$  składają się ze znaków angielskiego alfabetu. W definicji przesunięcia wprowadziliśmy  $\mathbb{Z}_{26}$ , ponieważ szyfrowanie i deszyfrowanie polegały na zastosowaniu pewnych działań algebraicznych. Stosując jednak szyfr podstawieniowy, wygodniej myśleć o szyfrowaniu i deszyfrowaniu jako o permutacjach zbioru znaków alfabetu.

Oto przykład „losowej” permutacji  $\pi$ , która mogłaby stanowić funkcję szyfrującą (jak poprzednio, małe litery oznaczają znaki tekstu jawnego, wielkie litery – znaki tekstu zaszyfrowanego).

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>	<i>h</i>	<i>i</i>	<i>j</i>	<i>k</i>	<i>l</i>	<i>m</i>
<i>X</i>	<i>N</i>	<i>Y</i>	<i>A</i>	<i>H</i>	<i>P</i>	<i>O</i>	<i>G</i>	<i>Z</i>	<i>Q</i>	<i>W</i>	<i>B</i>	<i>T</i>
<i>n</i>	<i>o</i>	<i>p</i>	<i>q</i>	<i>r</i>	<i>s</i>	<i>t</i>	<i>u</i>	<i>v</i>	<i>Ww</i>	<i>x</i>	<i>y</i>	<i>z</i>
<i>S</i>	<i>F</i>	<i>L</i>	<i>R</i>	<i>C</i>	<i>V</i>	<i>M</i>	<i>U</i>	<i>E</i>	<i>K</i>	<i>J</i>	<i>D</i>	<i>I</i>

I tak,  $e_\pi(a) = X$ ,  $e_\pi(b) = N$  itd. Funkcją deszyfrującą jest permutacja odwrotna, którą można otrzymać, zamieniając miejscami pierwszy i drugi wiersz,

a następnie porządkując alfabetycznie litery z pierwszego wiersza. Powstaje wtedy następująca tabelka:

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>	<i>H</i>	<i>I</i>	<i>J</i>	<i>K</i>	<i>L</i>	<i>M</i>
<i>d</i>	<i>l</i>	<i>r</i>	<i>y</i>	<i>v</i>	<i>o</i>	<i>h</i>	<i>e</i>	<i>z</i>	<i>x</i>	<i>w</i>	<i>p</i>	<i>t</i>
<i>N</i>	<i>O</i>	<i>P</i>	<i>Q</i>	<i>R</i>	<i>S</i>	<i>T</i>	<i>U</i>	<i>V</i>	<i>W</i>	<i>X</i>	<i>Y</i>	<i>Z</i>
<i>b</i>	<i>g</i>	<i>f</i>	<i>j</i>	<i>q</i>	<i>n</i>	<i>m</i>	<i>u</i>	<i>s</i>	<i>k</i>	<i>a</i>	<i>c</i>	<i>i</i>

Mamy zatem:  $d_\pi(A) = d$ ,  $d_\pi(B) = l$  itd.

Czytelnik jako ćwiczenie może spróbować odtajnić następujący tekst zaszyfrowany, używając określonej wyżej reguły deszyfrowania:

MGZVYZLGHCMHJMYXSSFMNHAYCDLMHA<sup>†</sup>.

Kluczem do szyfru podstawieniowego jest po prostu permutacja 26 znaków alfabetu. Takich permutacji jest  $26!$ , co jest liczbą ogromną, większą od  $4 \cdot 10^{26}$ . Metoda wyczerpującego przeszukiwania kluczy jest tu całkowicie bezużyteczna, nawet jeśli zaprzęgniemy do pracy komputer. Przekonamy się jednak później, że szyfr podstawieniowy można złamać innymi metodami.

### 1.1.3. Szyfr afiniczny

Szyfr z przesunięciem to szczególny przypadek szyfru podstawieniowego, w którym wykorzystuje się jedynie 26 spośród 26! możliwych permutacji 26 elementów. Innym szczególnym przypadkiem jest szyfr afiniczny, którym zajmiemy się w tym punkcie. W szyfrze afincznym ograniczamy klasę funkcji szyfrowania do funkcji postaci

$$e(x) = ax + b \pmod{26},$$

dla  $a, b \in \mathbb{Z}_{26}$ . Takie funkcje nazywa się *afinicznymi*, stąd też nazwa **szyfru afanicznego**. (Zauważmy, że gdy  $a = 1$ , mamy do czynienia z przesunięciem).

Aby umożliwić deszyfrowanie, musimy ustalić, kiedy funkcja afiniczna jest różnowartościowa. Inaczej mówiąc, chcemy, by dla dowolnego  $y \in \mathbb{Z}_{26}$  kongruencja

$$ax + b \equiv y \pmod{26}$$

miała dokładnie jedno rozwiązanie dla zmiennej  $x$ . Możemy tę kongruencję zapisać w postaci równoważnej:

$$ax \equiv y - b \pmod{26}.$$

Gdy wartości  $y$  przebiegają cały zbiór  $\mathbb{Z}_{26}$ , odpowiadające im wartości  $y - b$  także wyczerpują  $\mathbb{Z}_{26}$ . Stąd wystarczy zbadać kongruencję  $ax \equiv y \pmod{26}$  ( $y \in \mathbb{Z}_{26}$ ).

---

<sup>†</sup>Otrzymany tekst jawny oznacza: „tego tekstu zaszyfrowanego nie można odszyfrować” (przyp. tłum.).

Twierdzimy, że kongruencja ta ma dla każdego  $y$  dokładnie jedno rozwiązanie wtedy i tylko wtedy, gdy  $\text{NWD}(a, 26) = 1$  (gdzie NWD oznacza największy wspólny dzielnik dwóch liczb). Przypuśćmy bowiem, że  $\text{NWD}(a, 26) = d > 1$ . Wtedy kongruencja  $ax \equiv 0 \pmod{26}$  ma (co najmniej) dwa rozwiązania w  $\mathbb{Z}_{26}$ , mianowicie:  $x = 0$  oraz  $x = 26/d$ . W tym przypadku  $e(x) = ax + b \pmod{26}$  nie jest funkcją różnowartościową, nie nadaje się więc do użycia jako funkcja szyfrująca.

Na przykład, z tego, że  $\text{NWD}(4, 26) = 2$ , wynika, że funkcja  $4x + 7$  nie jest poprawną funkcją szyfrującą; dla każdego  $x \in \mathbb{Z}_{26}$  liczby  $x$  i  $x + 13$  otrzymałyby po zaszyfrowaniu tę samą wartość.

Załóżmy teraz, że  $\text{NWD}(a, 26) = 1$  i przypuśćmy, że dla pewnych liczb  $x_1$  i  $x_2$  mamy

$$ax_1 \equiv ax_2 \pmod{26}.$$

Wtedy

$$a(x_1 - x_2) \equiv 0 \pmod{26},$$

a stąd

$$26|a(x_1 - x_2).$$

Skorzystajmy teraz z następującej własności dzielenia: jeśli  $\text{NWD}(a, b) = 1$  oraz  $a|bc$ , to  $a|c$ . Ponieważ  $26|a(x_1 - x_2)$  oraz  $\text{NWD}(a, 26) = 1$ , musi zatem być spełniony warunek

$$26|(x_1 - x_2),$$

a w konsekwencji  $x_1 \equiv x_2 \pmod{26}$ .

Wykazaliśmy dotąd, że jeśli  $\text{NWD}(a, 26) = 1$ , to kongruencja postaci  $ax \equiv y \pmod{26}$  ma co najwyżej jedno rozwiązanie w  $\mathbb{Z}_{26}$ . Tak więc, gdy w  $ax \pmod{26}$  będziemy podstawiać za  $x$  wszystkie możliwe wartości z  $\mathbb{Z}_{26}$ , otrzymamy 26 różnych wartości modulo 26. Inaczej mówiąc, każda wartość będzie przyjmowana dokładnie raz. Oznacza to, że dla dowolnego  $y \in \mathbb{Z}_{26}$ , kongruencja  $ax \equiv y \pmod{26}$  ma dokładnie jedno rozwiązanie dla  $x$ .

Liczba 26 nie odgrywa w tym rozumowaniu żadnej szczególnej roli. Następujący wynik można udowodnić w podobny sposób.

### **TWIERDZENIE 1.1**

Kongruencja  $ax \equiv b \pmod{m}$  ma dokładnie jedno rozwiązanie  $x \in \mathbb{Z}_m$  dla każdego  $b \in \mathbb{Z}_m$  wtedy i tylko wtedy, gdy  $\text{NWD}(a, m) = 1$ .

Ponieważ  $26 = 2 \cdot 13$ , wartości  $a \in \mathbb{Z}_{26}$ , dla których  $\text{NWD}(a, 26) = 1$ , są następujące: 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23 oraz 25. Parametr  $b$  może być dowolnym elementem  $\mathbb{Z}_{26}$ . Stąd w szyfrze afincznym mamy do dyspozycji  $12 \cdot 26 = 312$  możliwych kluczy. (To, rzecz jasna, jest stanowczo za mało, by zapewnić bezpieczeństwo).

Rozważmy teraz sytuację ogólniejszą z modelem  $m$ . Potrzebujemy jeszcze jednego pojęcia z zakresu teorii liczb.

**DEFINICJA 1.3**

Niech  $a \geq 1$  i  $m \geq 2$  będą liczbami całkowitymi. Jeśli  $\text{NWD}(a, m) = 1$ , to mówimy, że liczby  $a$  i  $m$  są *względnie pierwsze*. Liczbę liczb całkowitych ze zbioru  $\mathbb{Z}_m$ , które są względnie pierwsze z  $m$ , oznacza się często symbolem  $\phi(m)$  (a funkcję  $\phi$  nazywa się *funkcją Eulera*).

Znany wynik z teorii liczb określa wartości  $\phi(m)$  w zależności od rozkładu  $m$  na czynniki pierwsze. (Liczba całkowita  $p > 1$  jest *pierwsza*, jeśli jej jedynymi dzielnikami dodatnimi są 1 i  $p$ ). Każdą liczbę całkowitą  $m > 1$  można jednoznacznie<sup>†</sup> rozłożyć na iloczyn potęg liczb pierwszych. Na przykład,  $60 = 2^2 \cdot 3 \cdot 5$  oraz  $98 = 2 \cdot 7^2$ .

Następne twierdzenie zawiera jawną wzór na wartość  $\phi(m)$ .

**TWIERDZENIE 1.2**

Niech

$$m = \prod_{i=1}^n p_i^{e_i},$$

gdzie  $p_i$  są różnymi liczbami pierwszymi oraz  $e_i > 0$ ,  $1 \leq i \leq n$ . Wówczas

$$\phi(m) = \prod_{i=1}^n (p_i^{e_i} - p_i^{e_i-1}).$$

Wynika stąd, że liczba kluczy dla szyfru aficznego nad  $\mathbb{Z}_m$  jest równa  $m\phi(m)$ , gdzie  $\phi(m)$  jest dane powyższym wzorem. (Dla funkcji szyfrowania posiadającej  $e(x) = ax + b$  mamy  $m$  możliwych wyborów liczby  $b$  oraz  $\phi(m)$  wyborów liczby  $a$ ). Na przykład, dla  $m = 60$ ,  $\phi(60) = 2 \cdot 2 \cdot 4 = 16$ , zatem liczba kluczy dla szyfru aficznego jest równa 960.

Rozważmy teraz operację deszyfrowania szyfru aficznego z modelem  $m = 26$ . Założymy, że  $\text{NWD}(a, 26) = 1$ . Przed przystąpieniem do deszyfrowania musimy rozwiązać kongruencję  $y \equiv ax + b \pmod{26}$  ze względu na  $x$ . Z wcześniejszych rozważań wiemy, że istnieje dokładnie jedno rozwiązanie w  $\mathbb{Z}_{26}$ , nie poznaliśmy jednak dotąd żadnej skutecznej metody znajdowania tego rozwiązania. Przydałby się do tego efektywny algorytm. Na szczęście taki algorytm deszyfrowania wynika z dalszych własności arytmetyki reszt.

Potrzebujemy pojęcia elementu odwrotnego.

**DEFINICJA 1.4**

Niech  $a \in \mathbb{Z}_m$ . Elementem odwrotnym do elementu  $a$  nazywamy element  $a^{-1} \in \mathbb{Z}_m$ , taki że  $aa^{-1} \equiv a^{-1}a \equiv 1 \pmod{m}$ .

Rozumowanie podobne do tego, które zastosowaliśmy wyżej, pozwala wykazać, że  $a$  ma element odwrotny modulo  $m$  wtedy i tylko wtedy, gdy

---

<sup>†</sup>Z dokładnością do kolejności czynników (przyp. tłum.).

$\text{NWD}(a, m) = 1$ ; zatem jeśli element odwrotny do  $a$  istnieje, to jest jedynym takim elementem. Zauważmy także, iż jeśli  $b = a^{-1}$ , to  $a = b^{-1}$ . Zauważmy też, że jeśli  $p$  jest liczbą pierwszą, to każdy niezerowy element  $\mathbb{Z}_p$  ma element odwrotny. Pierścień, w którym tak jest<sup>†</sup>, nazywamy *ciałem*.

W dalszej części książki opiszemy efektywny algorytm obliczania elementów odwrotnych w  $\mathbb{Z}_m$  dla dowolnej liczby  $m$ . Tymczasem w  $\mathbb{Z}_{26}$  metoda prób i błędów wystarcza do znalezienia takich elementów dla liczb względnie pierwszych z 26:  $1^{-1} = 1$ ,  $3^{-1} = 9$ ,  $5^{-1} = 21$ ,  $7^{-1} = 15$ ,  $11^{-1} = 19$ ,  $17^{-1} = 23$  i  $25^{-1} = 1$ . (Wszystkie te równości można łatwo sprawdzić. Na przykład,  $7 \cdot 15 = 105 \equiv 1 \pmod{26}$ , zatem  $7^{-1} = 15$ ).

Wróćmy do kongruencji  $y \equiv ax + b \pmod{26}$ . Jest ona równoważna kongruencji

$$ax \equiv y - b \pmod{26}.$$

Jeśli  $\text{NWD}(a, 26) = 1$ , to  $a$  ma element odwrotny modulo 26. Mnożąc obie strony kongruencji przez  $a^{-1}$ , otrzymujemy warunek

$$a^{-1}(ax) \equiv a^{-1}(y - b) \pmod{26}.$$

Skorzystajmy z łączności mnożenia modulo 26:

$$a^{-1}(ax) \equiv (a^{-1}a)x \equiv 1x \equiv x \pmod{26}.$$

W rezultacie  $x \equiv a^{-1}(y - b) \pmod{26}$ . To jest właśnie jawny wzór na  $x$ . Innymi słowy, funkcja deszyfrująca ma postać następującą:

$$d(y) = a^{-1}(y - b) \pmod{26}.$$

Pełny opis szyfru afinicznego znajduje się na rysunku 1.4. Popatrzmy na prosty przykład.

Niech  $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}$  i niech

$$\mathcal{K} = \{(a, b) \in \mathbb{Z}_{26} \times \mathbb{Z}_{26} : \text{NWD}(a, 26) = 1\}.$$

Dla  $K = (a, b) \in \mathcal{K}$  definiujemy

$$e_K(x) = ax + b \pmod{26}$$

oraz

$$d_K(y) = a^{-1}(y - b) \pmod{26}$$

$$x, y \in \mathbb{Z}_{26}.$$

#### RYSUNEK 1.4. Szyfr affine

---

<sup>†</sup>Czyli taki, w którym każdy niezerowy element ma element odwrotny (przyp. tłum.).

**Przykład 1.3**

Przyjmijmy, że  $K = (7, 3)$ . Jak stwierdziliśmy wyżej,  $7^{-1} \bmod 26 = 15$ . Funkcją szyfrującą jest funkcja

$$e_K(x) = 7x + 3,$$

odpowiadająca jej funkcja deszyfrująca wygląda tak:

$$d_K(y) = 15(y - 3) = 15y - 19,$$

gdzie wszystkie działania są wykonywane w  $\mathbb{Z}_{26}$ . Dobrym ćwiczeniem będzie sprawdzenie, że  $d_K(e_K(x)) = x$  dla każdego  $x \in \mathbb{Z}_{26}$ . Zgodnie z regułami obojętymi w  $\mathbb{Z}_{26}$  otrzymujemy:

$$\begin{aligned} d_K(e_K(x)) &= d_K(7x + 3) \\ &= 15(7x + 3) - 19 \\ &= x + 45 - 19 \\ &= x. \end{aligned}$$

Dla ilustracji zaszyfrujmy tekst jawnny *hot*. Zaczynamy od zamiany liter *h*, *o*, *t* na reszty modulo 26. Są to, odpowiednio, liczby 7, 14 i 19. Teraz szyfrujemy:

$$7 \cdot 7 + 3 \bmod 26 = 52 \bmod 26 = 0$$

$$7 \cdot 14 + 3 \bmod 26 = 101 \bmod 26 = 23$$

$$7 \cdot 19 + 3 \bmod 26 = 136 \bmod 26 = 6.$$

Tak więc mamy trzy znaki tekstu zaszyfrowanego: 0, 23 i 6, którym odpowiada ciąg liter *AXG*. Deszyfrowanie pozostawiamy Czytelnikowi jako ćwiczenie. □

**1.1.4. Szyfr Vigenèra'a**

Zarówno w szyfrze z przesunięciem, jak i w szyfrze afincznym, po ustaleniu klucza każdemu znakowi alfabetu tekstu jawnego odpowiada jednoznacznie wyznaczony znak alfabetu tekstu zaszyfrowanego. Z tego względu takie systemy kryptograficzne określa się jako *monoalfabetyczne*. Na rysunku 1.5 jest przedstawiony system kryptograficzny, który taki nie jest – dobrze znany **szyfr Vigenèra'a**. Nazwa pochodzi od nazwiska Blaise'a de Vigenèra'a, żyjącego w XVI wieku.

Odwołując się do opisanej wcześniej odpowiedniości  $A \leftrightarrow 0$ ,  $B \leftrightarrow 1, \dots$ ,  $Z \leftrightarrow 25$ , możemy z każdym kluczem  $K$  powiązać ciąg znaków alfabetu o długości  $m$ , zwany  *słowem kluczowym*. Szyfr Vigenèra'a koduje jednocześnie  $m$  znaków: każdy element tekstu jawnego odpowiada  $m$  znakom alfabetu.

Zobaczmy to na przykładzie.

Niech  $m$  będzie ustaloną dodatnią liczbą całkowitą i  $\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbb{Z}_{26})^m$ . Dla klucza  $K = (k_1, k_2, \dots, k_m)$  definiujemy

$$e_K(x_1, x_2, \dots, x_m) = (x_1 + k_1, x_2 + k_2, \dots, x_m + k_m)$$

oraz

$$d_K(y_1, y_2, \dots, y_m) = (y_1 - k_1, y_2 - \overbrace{k_2, \dots, y_m - k_m}),$$

przy czym wszystkie działania są wykonywane w  $\mathbb{Z}_{26}$ .

### RYSUNEK 1.5. Szyfr Vigenère'a

#### Przykład 1.4

Przyjmijmy  $m = 6$  i niech słowem kluczowym będzie *CIPHER*. Odpowiedniem liczbowym tego słowa jest  $K = (2, 8, 15, 7, 4, 17)$ . Przypuśćmy, że tekstem jawnym jest ciąg

thiscryptosystemisnotsecure<sup>†</sup>

Przekształcamy symbole tekstu jawnego w reszty modulo 26, zapisujemy je w grupach po 6 znaków, po czym „dodajemy” słowo kluczowe modulo 26:

19	7	8	18	2	17	24	15	19	14	18	24
2	8	15	7	4	17	2	8	15	7	4	17
21	15	23	25	6	8	0	23	8	21	22	15
18	19	4	12	8	18	13	14	19	18	4	2
2	8	15	7	4	17	2	8	15	7	4	17
20	1	19	19	12	9	15	22	8	25	8	19
20	17	4									
2	8	15									
22	25	19									

Otrzymanemu ciągowi liczb odpowiada następujący ciąg liter:

VPXZGIAIXIVWPUBTMJPWIZITWZT.

Przy deszyfrowaniu używamy tego samego słowa kluczowego, ale tym razem zamiast je dodawać, musimy je odjąć modulo 26.  $\square$

Zauważmy, że w szyfrze Vigenère'a mamy  $26^m$  możliwych słów kluczowych o długości  $m$ , zatem nawet dla stosunkowo małych wartości  $m$  wyczerpujące

<sup>†</sup>W tłumaczeniu „ten system kryptograficzny nie jest bezpieczny” (przyp. tłum.).

przeszukiwanie kluczy trwałoby za dugo. Na przykład, jeśli przyjmujemy  $m = 5$ , przestrzeń kluczy składa się z ponad  $1,1 \cdot 10^7$  elementów. To już wystarczająco dużo, by wykluczyć ręczne przeszukiwanie (choć nie komputerowe).

W szyfrze Vigenèra' ze słowem kluczowym o długości  $m$  znak alfabetu może być przekształcony na jeden z  $m$  możliwych znaków (zakładając, że wszystkie znaki słowa kluczowego są różne). Taki system kryptograficzny nosi miano *polialfabetycznego*<sup>†</sup>. Na ogół kryptoanaliza systemów polialfabetycznych jest trudniejsza niż w przypadku systemów monoalfabetycznych.

### 1.1.5. Szyfr Hilla

Opiszymy tu inny system polialfabetyczny, zwany **szyfrem Hilla**. Został on wymyślony w 1929 r. przez Lestera S. Hilla. Niech  $m$  będzie dodatnią liczbą całkowitą i niech  $\mathcal{P} = \mathcal{C} = (\mathbb{Z}_{26})^m$ . Pomysł polega na tym, by wziąć  $m$  kombinacji liniowych  $m$  znaków alfabetu występujących w jednym elemencie tekstu jawnego, otrzymując w ten sposób  $m$  znaków występujących w jednym elemencie tekstu zaszyfrowanego.

Na przykład, gdy  $m = 2$ , możemy zapisać element tekstu jawnego w postaci  $x = (x_1, x_2)$ , natomiast element tekstu zaszyfrowanego jako  $y = (y_1, y_2)$ , gdzie  $y_1$  i  $y_2$  są kombinacjami liniowymi znaków  $x_1$  i  $x_2$ . Możemy przyjąć, że

$$y_1 = 11x_1 + 3x_2$$

$$y_2 = 8x_1 + 7x_2.$$

Można to oczywiście przedstawić zwięźlej w zapisie macierzowym:

$$(y_1, y_2) = (x_1, x_2) \begin{pmatrix} 11 & 3 \\ 8 & 7 \end{pmatrix}.$$

Ogólniej, kluczem będzie macierz  $K$  o  $m$  wierszach i  $m$  kolumnach ( $m \times m$ ). Jeśli element występujący w wierszu o numerze  $i$  i kolumnie o numerze  $j$  macierzy  $K$  nazwiemy  $k_{i,j}$ , to piszemy  $K = (k_{i,j})$ . Dla  $x = (x_1, \dots, x_m) \in \mathcal{P}$  i  $K \in \mathcal{K}$  obliczamy  $y = e_K(x) = (y_1, \dots, y_m)$  w następujący sposób:

$$(y_1, \dots, y_m) = (x_1, \dots, x_m) \begin{pmatrix} k_{1,1} & k_{1,2} & \dots & k_{1,m} \\ k_{2,1} & k_{2,2} & \dots & k_{2,m} \\ \vdots & \vdots & & \vdots \\ k_{m,1} & k_{m,2} & \dots & k_{m,m} \end{pmatrix}.$$

Innymi słowy,  $y = xK$ .

Mówimy, że tekst zaszyfrowany powstał z tekstu jawnego przez *przekształcenie liniowe*. Pozostaje rozważyć, jak będzie przebiegało deszyfrowanie, czyli jak odzyskać  $x$ , znając  $y$ . Czytelnik znający algebrę liniową domyśli się, że do deszyfrowania użyjemy macierzy odwrotnej  $K^{-1}$ . Tekst zaszyfrowany odczytujemy za pomocą wzoru  $x = yK^{-1}$ .

---

<sup>†</sup>Spotyka się też nazwę wieloalfabetowy (przyp. tłum.).

Przypomnijmy niezbędne pojęcia algebry liniowej. Jeśli  $A = (a_{i,j})$  jest macierzą  $l \times m$ , a  $B = (b_{j,k})$  macierzą  $m \times n$ , to *iloczyn macierzy*  $AB = (c_{i,k})$  określmy wzorem

$$c_{i,k} = \sum_{j=1}^m a_{i,j} b_{j,k}$$

dla  $1 \leq i \leq l$  i  $1 \leq k \leq n$ . Oznacza to, że element występujący w macierzy  $AB$  w wierszu  $i$  i w kolumnie  $k$  powstaje w wyniku wymnożenia kolejnych elementów wiersza  $i$  macierzy  $A$  przez odpowiednie elementy kolumny  $j$  macierzy  $B$  i dodanie otrzymanych iloczynów. Zauważmy, że macierz  $AB$  jest macierzą  $l \times n$ .

Tak określone mnożenie macierzy jest łączne (to znaczy  $A(BC) = (AB)C$ ), ale na ogół nie jest przemienne (to znaczy nie zawsze  $AB = BA$ , nawet gdy macierze  $A$  i  $B$  są kwadratowe).

Macierzą jednostkową  $m \times m$ , oznaczaną symbolem  $I_m$ , jest macierz  $m \times m$ , która ma 1 na głównej przekątnej i 0 w pozostałych miejscach. Na przykład, macierz jednostkowa  $2 \times 2$  to macierz

$$I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Macierz  $I_m$  nazywa się macierzą jednostkową, gdyż  $AI_m = A$  dla dowolnej macierzy  $A$  o  $l$  wierszach i  $m$  kolumnach oraz  $I_mB = B$  dla dowolnej macierzy  $B$  o  $m$  wierszach i  $n$  kolumnach. Jeśli  $A$  jest macierzą  $m \times m$ , to macierzą *odwrotną* do niej nazywamy macierz  $A^{-1}$ , taką że  $AA^{-1} = A^{-1}A = I_m$  (jeśli taka macierz istnieje). Nie dla każdej macierzy istnieje macierz odwrotna, ale jeśli istnieje, to tylko jedna.

Dysponując tymi własnościami macierzy, bez trudu wyrowadzimy podany wyżej wzór na deszyfrowanie: ponieważ  $y = xK$ , możemy pomnożyć obie strony tej równości przez macierz  $K^{-1}$ , otrzymując równości następujące:

$$yK^{-1} = (xK)K^{-1} = x(KK^{-1}) = xI_m = x.$$

(Zauważmy wykorzystanie łączności w tych przekształceniach).

Sprawdźmy, że macierz szyfrowania ma macierz odwrotną w  $\mathbb{Z}_{26}$ :

$$\begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}^{-1} = \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix},$$

ponieważ

$$\begin{aligned} \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix} &= \begin{pmatrix} 11 \cdot 7 + 8 \cdot 23 & 11 \cdot 18 + 8 \cdot 11 \\ 3 \cdot 7 + 7 \cdot 23 & 3 \cdot 18 + 7 \cdot 11 \end{pmatrix} \\ &= \begin{pmatrix} 261 & 286 \\ 182 & 131 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \end{aligned}$$

(Przypomnijmy, że działania arytmetyczne wykonujemy modulo 26).

Rozważmy teraz przykład ilustrujący szyfrowanie i deszyfrowanie w szyfrze Hilla.

### Przykład 1.5

Przypuśćmy, że kluczem jest macierz

$$K = \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}.$$

Z wcześniejszych obliczeń wiemy, że

$$K^{-1} = \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix}.$$

Niech tekstem do utajnienia będzie słowo *july*. Mamy do zaszyfrowania dwa elementy tekstu jawnego: (9, 20) (odpowiadający znakom *ju*) oraz (11, 24) (odpowiadający *ly*). Obliczamy:

$$(9, 20) \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} = (99 + 60, 72 + 140) = (3, 4)$$

oraz

$$(11, 24) \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} = (121 + 72, 88 + 168) = (11, 22).$$

W rezultacie słowo *july* zostało zaszyfrowane jako *DELW*. Przystępując do deszyfrowania, Bolek wykona następujące obliczenia:

$$(3, 4) \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix} = (9, 20)$$

oraz

$$(11, 22) \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix} = (11, 24),$$

otrzymując poprawny tekst jawny. □

Wykazaliśmy, jak dotąd, że deszyfrowanie jest możliwe, gdy  $K$  ma macierz odwrotną. W istocie  $K$  musi mieć macierz odwrotną, aby można było poprawnie deszyfrować. (Wynika to dość łatwo z elementarnej alebry liniowej, ale nie będziemy tego tu dowodzić). Interesują nas zatem tylko takie macierze  $K$ , które są odwracalne.

Odwracalność macierzy (kwadratowej) zależy od wartości jej wyznacznika. Aby uniknąć nadmiernych uogólnień, skupimy się wyłącznie na przypadku  $2 \times 2$ .

**DEFINICJA 1.5**

Wyznacznikiem macierzy  $A = (a_{i,j})$  o 2 wierszach i 2 kolumnach jest liczba

$$\det A = a_{1,1}a_{2,2} - a_{1,2}a_{2,1}.$$

**UWAGA** Wyznacznik macierzy kwadratowej  $m \times m$  można obliczyć za pomocą elementarnych działań na wierszach; niezbędne informacje można znaleźć w każdym podręczniku algebry liniowej. ■

Przypomnijmy dwie ważne własności wyznaczników: równość  $\det I_m = 1$  oraz prawo mnożenia:  $\det(AB) = \det A \cdot \det B$ .

Macierz o wyrazach rzeczywistych ma macierz odwrotną wtedy i tylko wtedy, gdy jej wyznacznik jest różny od zera. Pamiętajmy jednak, że teraz działamy w  $\mathbb{Z}_{26}$ , zatem istotny dla naszych celów warunek ma inną postać: macierz  $K$  ma macierz odwrotną modulo 26 wtedy i tylko wtedy, gdy  $\text{NWD}(\det K, 26) = 1$ .

Naszkicujemy w skrócie dowód tego faktu. Najpierw założymy, że  $\text{NWD}(\det K, 26) = 1$ . Wówczas dla liczby  $\det K$  istnieje w  $\mathbb{Z}_{26}$  liczba odwrotna. Dalej, dla  $1 \leq i \leq m$ ,  $1 \leq j \leq m$  określmy macierz  $K_{ij}$  jako macierz otrzymaną z  $K$  przez usunięcie wiersza o numerze  $i$  i kolumny o numerze  $j$ . Z kolei niech  $K^*$  będzie macierzą, która na miejscu  $(i, j)$  ma wartość  $(-1)^{i+j} \det K_{ij}$ . (Macierz  $K^*$  nazywa się *macierzą sprzężoną do macierzy  $K$* ). Można wtedy wykazać, że

$$K^{-1} = (\det K)^{-1} K^*.$$

Tak więc  $K$  jest odwracalna.

Założymy na odwrot, że  $K$  ma macierz odwrotną  $K^{-1}$ . Z prawa mnożenia dla wyznaczników wnioskujemy, że

$$1 = \det I = \det(KK^{-1}) = \det K \det K^{-1}.$$

To dowodzi, że liczba  $\det K$  jest odwracalna w  $\mathbb{Z}_{26}$ .

**UWAGA** Powyższy wzór na macierz  $K^{-1}$  jest obliczeniowo efektywny tylko dla małych wartości  $m$  (powiedzmy,  $m = 2, 3$ ). Dla większych  $m$  metoda obliczania macierzy odwrotnych wymaga stosowania elementarnych operacji na wierszach. ■

W przypadku macierzy  $2 \times 2$  mamy następujący wzór:

**TWIERDZENIE 1.3**

Niech  $A = (a_{i,j})$  będzie macierzą  $2 \times 2$  nad  $\mathbb{Z}_{26}$ , taką że liczba  $\det A = a_{1,1}a_{2,2} - a_{1,2}a_{2,1}$  jest odwracalna. Wtedy

$$A^{-1} = (\det A)^{-1} \begin{pmatrix} a_{2,2} & -a_{1,2} \\ -a_{2,1} & a_{1,1} \end{pmatrix}.$$

Wróćmy do wcześniejszego przykładu. Po pierwsze,

$$\begin{aligned}\det \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} &= 11 \cdot 7 - 8 \cdot 3 \bmod 26 \\ &= 77 - 24 \bmod 26 \\ &= 53 \bmod 26 \\ &= 1.\end{aligned}$$

Dalej,  $1^{-1} \bmod 26 = 1$ , zatem macierz odwrotna wygląda tak:

$$\begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}^{-1} = \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix},$$

jak już sprawdziliśmy wcześniej.

Dokładny opis szyfru Hilla nad  $\mathbb{Z}_{26}$  znajduje się na rysunku 1.6.

Niech  $m$  będzie ustaloną dodatnią liczbą całkowitą i  $\mathcal{P} = \mathcal{C} = (\mathbb{Z}_{26})^m$ .

Niech ponadto

$$\mathcal{K} = \{\text{odwracalne macierze } m \times m \text{ nad } \mathbb{Z}_{26}\}.$$

Dla klucza  $K$  definiujemy

$$e_K(x) = xK$$

oraz

$$d_K(y) = yK^{-1},$$

przy czym wszystkie działania są wykonywane w  $\mathbb{Z}_{26}$ .

#### RYSUNEK 1.6. Szyfr Hilla

##### 1.1.6. Szyfr permutujący

Wszystkie systemy kryptograficzne, które rozpatrywaliśmy do tej pory, zakładały podstawienie: znaki tekstu jawnego były zastępowane przez różne znaki tekstu zaszyfrowanego. Istotą szyfru permutującego jest to, że znaki tekstu jawnego pozostają niezmienne, zmienia się jedynie ich rozmieszczenie w tekście. **Szyfr permutujący** (zwany także **szyfrem przestawieniowym**) jest znany i używany od wielu stuleci. Już w 1563 roku Giovanni Porta wskazywał na różnice między szyfrem permutującym a szyfrem podstawieniowym. Formalną definicję podajemy na rysunku 1.7.

Podobnie jak w przypadku szyfru podstawieniowego, wygodniej jest operować znakami alfabetu zamiast resztami modulo 26, nie wykonuje się tu bowiem żadnych operacji algebraicznych podczas szyfrowania lub deszyfrowania.

Niech  $m$  będzie ustaloną dodatnią liczbą całkowitą i  $\mathcal{P} = \mathcal{C} = (\mathbb{Z}_{26})^m$ . Niech ponadto  $\mathcal{K}$  będzie zbiorem wszystkich permutacji zbioru  $\{1, \dots, m\}$ . Dla klucza (czyli permutacji)  $\pi$  definiujemy

$$e_\pi(x_1, \dots, x_m) = (x_{\pi(1)}, \dots, x_{\pi(m)})$$

oraz

$$d_K(y_1, \dots, y_m) = (y_{\pi^{-1}(1)}, \dots, y_{\pi^{-1}(m)}),$$

gdzie  $\pi^{-1}$  jest permutacją odwrotną do  $\pi$ .

### RYSUNEK 1.7. Szyfr permutujący

Oto przykład ilustrujący istotę rzeczy.

#### Przykład 1.6

Niech  $m = 6$  i niech kluczem będzie następująca permutacja  $\pi$ :

1	2	3	4	5	6
3	5	1	6	4	2

Wówczas permutacją odwrotną  $\pi^{-1}$  jest funkcja

1	2	3	4	5	6
3	6	1	5	2	4

Przypuśćmy, że tekstem jawnym jest zdanie

shesellsseashellsbytheseashore<sup>†</sup>.

Napierw łączymy znaki tekstu jawnego w grupy sześcioliterowe:

shesel | lsseas | hellsb | ythese | ashore

Teraz w każdej z tych grup przedstawiamy litery zgodnie z permutacją  $\pi$ . Otrzymujemy tekst następujący:

EESLSH | SALSES | LSHBLE | HSYEET | HRAEOS

W rezultacie tekst zaszyfrowany jest ciągiem

EESLSHSALSESLSHBLEHSYEETHRAEOS.

Tekst zaszyfrowany można teraz odczytać w podobny sposób, używając tym razem permutacji odwrotnej  $\pi^{-1}$ . □

<sup>†</sup> W tłumaczeniu: „Ona sprzedaje morskie muszle przy brzegu morza” – angielski odpowiednik polskiego „w czasie suszy szosa sucha” (przyp. tłum.).

Szyfr permutujący jest w istocie szczególnym przypadkiem szyfru Hilla. Dla danej permutacji  $\pi$  zbioru  $\{1, \dots, m\}$  możemy określić związaną z nią macierz permutacji  $K_\pi = (k_{i,j})$  o  $m$  wierszach i  $m$  kolumnach zgodnie ze wzorem

$$k_{i,j} = \begin{cases} 1, & \text{gdy } i = \pi(j), \\ 0, & \text{w przeciwnym razie.} \end{cases}$$

(Macierz permutacji to macierz, która w każdym wierszu i w każdej kolumnie ma dokładnie raz „1”, a na pozostałych miejscach „0”. Każdą macierz permutacji można otrzymać z macierzy jednostkowej przez permutację wierszy lub kolumn).

Nietrudno zauważyć, że szyfrowanie metodą Hilla za pomocą macierzy  $K_\pi$  jest tym samym, co szyfrowanie permutujące z użyciem permutacji  $\pi$ . Ponadto,  $K_\pi^{-1} = K_{\pi^{-1}}$ , a więc macierz odwrotna do macierzy  $K_\pi$  jest macierzą permutacji odpowiadającą permutacji odwrotnej  $\pi^{-1}$ . Stąd szyfrowanie Hilla jest równoważne szyfrowaniu permutującemu.

Macierzami permutacji wyznaczonymi przez permutację  $\pi$  z ostatniego przykładu są macierze

$$K_\pi = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

oraz

$$K_\pi^{-1} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Czytelnik może sprawdzić, że iloczyn tych dwóch macierzy jest macierzą jednostkową.

### 1.1.7. Szyfry strumieniowe

W systemach kryptograficznych, które badaliśmy do tej pory, kolejne elementy tekstu jawnego są szyfrowane tym samym kluczem. Inaczej mówiąc, ciąg tekstu zaszyfrowanego powstaje w sposób następujący:

$$\mathbf{y} = y_1 y_2 \dots = e_K(x_1) e_K(x_2) \dots$$

Systemy tego typu są często nazywane *szyframi blokowymi*.

Szyfry strumieniowe reprezentują inne podejście. Ich istota polega na tym, że tworzy się klucz strumieniowy  $\mathbf{z} = z_1 z_2 \dots$  i używa się go do szyfrowania strumienia tekstu jawnego  $\mathbf{x} = x_1 x_2 \dots$  zgodnie z następującą zasadą:

$$\mathbf{y} = y_1 y_2 \dots = e_{x_1}(x_1) e_{x_2}(x_2) \dots$$

A oto jak działa szyfr strumieniowy. Założmy, że  $K \in \mathcal{K}$  jest kluczem, a  $x_1x_2\dots$  strumieniem (ciągiem) tekstu jawnego. Do generowania  $z_i$  ( $i$ -tego elementu w strumieniu klucza) stosuje się funkcję  $f_i$ , której wartość zależy od klucza  $K$  oraz od pierwszych  $i - 1$  znaków tekstu jawnego:

$$z_i = f_i(K, x_1, x_2, \dots, x_{i-1}).$$

Elementu  $z_i$  klucza strumieniowego używa się do szyfrowania  $x_i$ , w wyniku czego otrzymujemy się  $y_i = e_{z_i}(x_i)$ . Tak więc podczas szyfrowania tekstu jawnego  $x_1x_2\dots$  oblicza się kolejno

$$z_1, y_1, z_2, y_2, \dots$$

Deszyfrowanie ciągu  $y_1y_2\dots$  wymaga kolejnego obliczenia

$$z_1, x_1, z_2, x_2, \dots$$

Przedstawimy teraz formalną definicję matematyczną.

### DEFINICJA 1.6

Szyfrem strumieniowym nazywamy układ  $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{L}, \mathcal{F}, \mathcal{E}, \mathcal{D})$  spełniający następujące warunki:

- (1)  $\mathcal{P}$  jest skończonym zbiorem możliwych *tekstów jawnych*.
- (2)  $\mathcal{C}$  jest skończonym zbiorem możliwych *tekstów zaszyfrowanych*.
- (3)  $\mathcal{K}$ , *przestrzeń kluczy*, jest skończonym zbiorem możliwych kluczy.
- (4)  $\mathcal{L}$  jest skończonym zbiorem, zwanym *alfabetem klucza strumieniowego*.
- (5)  $\mathcal{F} = (f_1, f_2, \dots)$  jest *generatorem klucza strumieniowego*. Dla  $i \geq 1$   

$$f_i: \mathcal{K} \times \mathcal{P}^{i-1} \rightarrow \mathcal{L}.$$
- (6) Dla każdego  $z \in \mathcal{L}$  istnieje *reguła szyfrowania*  $e_z \in \mathcal{E}$  oraz odpowiadająca jej *reguła deszyfrowania*  $d_z \in \mathcal{D}$ . Reguły  $e_z: \mathcal{P} \rightarrow \mathcal{C}$  i  $d_z: \mathcal{C} \rightarrow \mathcal{P}$  są funkcjami, takimi że  $d_z(e_z(x)) = x$  dla każdego tekstu jawnego  $x \in \mathcal{P}$ .

Szyfr blokowy można traktować jako szczególny przypadek szyfru strumieniowego, przyjmując stałą wartość strumienia kluczy:  $z_i = K$  dla wszystkich  $i \geq 1$ .

Oto niektóre rodzaje szyfrów strumieniowych wraz z poglądowymi przykładami. *Synchroniczny* szyfr strumieniowy to taki szyfr strumieniowy, w którym strumień kluczy nie zależy od tekstu jawnego. Strumień powstaje tu jako funkcja tylko jednej zmiennej  $K$ . W takim przypadku możemy myśleć o  $K$  jako o „zalążku”, z którego powstaje strumień kluczy  $z_1z_2\dots$

Z kolei szyfr strumieniowy jest *okresowy* z okresem  $d$ , gdy  $z_{i+d} = z_i$  dla wszystkich liczb całkowitych  $i \geq 1$ . Szyfr Vigenère'a ze słowem kluczowym o dłu-

gości  $m$  możemy traktować jako szyfr okresowy z okresem  $m$ . W tym przypadku kluczem jest  $K = (k_1, \dots, k_m)$ . Ciąg  $K$  dostarcza pierwszych  $m$  elementów klucza strumieniowego:  $z_i = k_i$  dla  $1 \leq i \leq m$ , po czym od tego miejsca klucz strumienia po prostu się powtarza. Zauważmy, że jeśli spojrzymy na szyfr Vigenère'a jako na szyfr strumieniowy, to funkcje szyfrująca i deszyfrująca pokrywają się z tymi, których użyliśmy w szyfrze z przesunięciem:  $e_z(x) = x + z$  oraz  $d_z(y) = y - z$ .

Szyfry strumieniowe są często opisywane za pomocą alfabetów binarnych:  $\mathcal{P} = \mathcal{C} = \mathcal{L} = \mathbb{Z}_2$ . Wówczas operacje szyfrowania i deszyfrowania stają się po prostu działaniami dodawania modulo 2:

$$e_z(x) = x + z \bmod 2$$

oraz

$$d_z(y) = y + z \bmod 2.$$

Jeśli przyjmiemy, że „1” odpowiada wartości boolowskiej „prawda”, a „0” wartości boolowskiej „fałsz”, to dodawanie modulo 2 można interpretować jako operację różnicę symetryczną<sup>†</sup>, a to oznacza, że implementacja sprzętowa szyfrowania i deszyfrowania jest bardzo efektywna.

Popatrzmy na inny sposób generowania (synchronicznego) klucza strumieniowego. Zacznijmy od  $(k_1, \dots, k_m)$  i niech  $z_i = k_i$  dla  $1 \leq i \leq m$  (jak poprzednio). Tym razem użyjmy jednak do utworzenia klucza strumieniowego liniowej rekurencji stopnia  $m$ :

$$z_{i+m} = \sum_{j=0}^{m-1} c_j z_{i+j} \bmod 2,$$

gdzie  $c_0, \dots, c_{m-1} \in \mathbb{Z}_2$  są ustalonymi stałymi.

**UWAGA** Mówimy tu o rekurencji *stopnia  $m$* , gdyż każdy wyraz strumienia zależy od  $m$  poprzednich wyrazów. Określamy ją zaś jako *liniową*, ponieważ  $z_{i+m}$  jest liniową funkcją wcześniejszych wyrazów. Zauważmy, że bez utraty ogólności możemy przyjąć  $c_0 = 1$ , w przeciwnym przypadku mielibyśmy bowiem do czynienia z rekurencją stopnia  $m - 1$ . ■

Klucz składa się tu z  $2m$  wartości  $k_1, \dots, k_m, c_0, \dots, c_{m-1}$ . Jeśli

$$(k_1, \dots, k_m) = (0, \dots, 0),$$

to ciąg składa się z samych zer. Tego, rzecz jasna, należy unikać, gdyż tekst zaszyfrowany byłby tożsamym z tekstem jawnym. Jeśli jednak dobierzemy odpowiednio

---

<sup>†</sup>angielski termin *exclusive-or* jest tłumaczony także jako alternatywa wykluczająca lub jako suma modulo 2 (przyp. tłum.).

stałe  $c_0, \dots, c_{m-1}$ , to z każdego innego wektora początkowego  $(k_1, \dots, k_m)$  powstanie okresowy klucz strumieniowy o okresie  $2^m - 1$ . Tak więc „krótki” klucz generuje klucz strumieniowy o bardzo długim okresie. To z pewnością sytuacja pożądana: zobaczymy dalej, jak można złamać szyfr Vigenère'a, wykorzystując fakt, że klucz strumieniowy ma mały okres.

Oto przykład, który to ilustruje.

### Przykład 1.7

Niech  $m = 4$ . Założymy, że klucz strumieniowy jest generowany według następującej zasady:

$$z_{i+4} = z_i + z_{i+1} \bmod 2$$

( $i \geq 1$ ). Jeśli zaczniemy od wektora różnego od  $(0, \dots, 0)$ , to otrzymamy klucz strumieniowy o okresie 15. Na przykład, przyjmując  $(1, 0, 0, 0)$  jako wektor początkowy, będziemy mieli następujący strumień:

$$1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, \dots$$

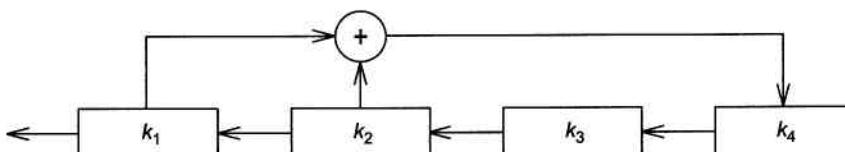
Każdy inny niezerowy wektor początkowy da w wyniku cykliczną permutację tego samego klucza strumieniowego.  $\square$

Opisywana tu metoda generowania strumienia kluczy ma dodatkową zaletę: można ją skutecznie realizować sprzętowo za pomocą *liniowego rejestru przesuwającego ze sprzężeniem zwrotnym* (ang. *linear feedback shift register – LFSR*). Użylibyśmy tu rejestru  $m$ -bitowego z wektorem początkowym  $(k_1, \dots, k_m)$ . W każdej jednostce czasu wykonywane byłyby równolegle następujące operacje:

- (1) wybranie  $k_1$  jako kolejnego bitu strumienia;
- (2) przesunięcie każdego z elementów  $k_2, \dots, k_m$  o jeden bit w lewo;
- (3) obliczenie „nowej” wartości  $k_m$  jako

$$\sum_{j=0}^{m-1} c_j k_{j+1}$$

(to jest właśnie „liniowe sprzężenie zwrotne”).



RYSUNEK 1.8. Liniowy rejestr przesuwający ze sprzężeniem zwrotnym

Niech  $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathcal{L} = \mathbb{Z}_{26}$ . Niech dalej  $z_1 = K$  oraz  $z_i = x_{i-1}$  ( $i \geq 2$ ). Dla  $0 \leq z \leq 25$  definiujemy

$$e_z(x) = x + z \bmod 26$$

oraz

$$d_z(y) = y - z \bmod 26$$

$$(x, y \in \mathbb{Z}_{26}).$$

### RYSUNEK 1.9. Szyfr z automatycznym generowaniem klucza

Zauważmy, że liniowe sprzężenie zwrotne realizuje się przez wybranie niektórych bitów rejestru (wskazanych przez stałe  $c_j$  o wartości „1”) i obliczenie różnicy symetrycznej. Jest to zilustrowane na rysunku 1.8, przedstawiającym rejestr generujący klucz strumieniowy z przykładu 1.7.

Z kolei rysunek 1.9 przedstawia niesynchroniczny szyfr strumieniowy znany jako **szyfr z automatycznym generowaniem klucza**<sup>†</sup>. Pochodzi on, jak się wydaje, od Vigenère'a.

Nazwa tego szyfru wzięła się stąd, że tekst jawnny jest w nim używany w roli klucza (oprócz pierwotnego „rozruchowego” klucza  $K$ ). Następujący przykład ilustruje to podejście.

#### Przykład 1.8

Niech  $K = 8$  i założymy, że tekstem jawnym jest ciąg

rendezvous.

Najpierw przekształcamy tekst jawnny w ciąg liczb:

17 4 13 3 4 25 21 14 20 18

Klucz strumieniowy wygląda tak:

8 17 4 13 3 4 25 21 14 20

Dodajemy teraz odpowiednie elementy i redukujemy modulo 26:

25 21 17 16 7 3 20 9 8 12

W wersji alfabetycznej tekst zaszyfrowany przybiera następującą postać

ZVRQHDUJIM.

---

<sup>†</sup>Spotyka się też nazwę szyfr z autokluczem (przyp. tłum.).

Zobaczmy teraz, jak Alicja deszyfuje ten tekst. Zaczyna od przekształcenia ciągu alfabetycznego w ciąg liczbowy:

25 21 17 16 7 3 20 9 8 12,

po czym przystępuje do obliczeń:

$$x_1 = d_8(25) = 25 - 8 \bmod 26 = 17.$$

Dalej,

$$x_2 = d_{17}(21) = 21 - 17 \bmod 26 = 4,$$

i podobnie z kolejnymi symbolami. Po otrzymaniu kolejnego znaku tekstu jawnego używa go także jako kolejnego elementu klucza strumieniowego.  $\square$

Oczywiście szyfr z automatycznym generowaniem klucza nie jest bezpieczny, mamy w nim bowiem do dyspozycji jedynie 26 różnych kluczy.

W następnym podrozdziale omówimy metody, które mogą posłużyć do kryptoanalizy poznanych dotąd systemów kryptograficznych.

---

## 1.2. Kryptoanaliza

Omówimy w tym podrozdziale niektóre metody kryptoanalizy. Przyjmujemy ogólne założenie, że przeciwnik, Oskar, wie jaki system kryptograficzny został zastosowany (co często bywa nazywane *zasadą Kerckhoffa*). Oczywiście, jeśli tak nie jest, jego zadanie staje się trudniejsze. Nie chcemy jednak opierać bezpieczeństwa systemu na niepewnej przesłance, iż Oskar nie ma żadnej wiedzy o użytym przez nas systemie. Tak więc przy projektowaniu bezpiecznego systemu kryptograficznego musimy uwzględnić zasadę Kerckhoffa.

Rozróżnijmy przede wszystkim metody ataku na system kryptograficzny. Przyjrzyjmy się najczęściej spotykany sytuacjom.

**Tylko tekst zaszyfrowany.** Przeciwnik jest w posiadaniu ciągu tekstu zaszyfrowanego  $y$ .

**Znany tekst jawny.** Przeciwnik jest w posiadaniu ciągu tekstu jawnego  $x$  wraz z odpowiadającym mu ciągiem tekstu zaszyfrowanego  $y$ .

**Wybrany tekst jawny.** Przeciwnik uzyskał tymczasowy dostęp do aparatury deszyfrującej, może więc wybrać ciąg tekstu jawnego  $x$  i zbudować odpowiadający mu ciąg tekstu zaszyfrowanego  $y$ .

**Wybrany tekst zaszyfrowany.** Przeciwnik uzyskał tymczasowy dostęp do aparatury deszyfrującej, może więc wybrać ciąg tekstu zaszyfrowanego  $y$  i zbudować odpowiadający mu ciąg tekstu jawnego.

W każdym przypadku zadanie przeciwnika polega na ustaleniu klucza użytego do szyfrowania. Zauważmy w tym miejscu, że atak z wybranym tekstem zaszyfrowanym ma szczególne znaczenie dla systemów kryptograficznych z kluczem publicznym, które omówimy w dalszych rozdziałach.

Rozważmy najpierw najsłabszy typ ataku, mianowicie atak tylko z tekstem zaszyfrowanym. Zakładamy, że tekst jawny jest zwyczajnym tekstem w języku angielskim<sup>†</sup>, bez znaków przestankowych i odstępów między wyrazami (co utrudnia kryptoanalizę).

W wielu metodach technik kryptoanalytycznych wykorzystuje się statystyczne właściwości języka angielskiego. Gromadząc dane statystyczne z licznych książek, czasopism i gazet, różni badacze szacowali względnączęstość występowania każdej z 26 liter angielskiego alfabetu. Oszacowania, które prezentujemy w tablicy 1.1, zostały uzyskane przez Bekera i Pipera.

**TABLICA 1.1. Prawdopodobieństwo wystąpienia 26 liter alfabetu języka angielskiego**

Litera	Prawdopodobieństwo	Litera	Prawdopodobieństwo
A	0,082	N	0,067
B	0,015	O	0,075
C	0,028	P	0,019
D	0,043	Q	0,001
E	0,127	R	0,060
F	0,022	S	0,063
G	0,020	T	0,091
H	0,061	U	0,028
I	0,070	V	0,010
J	0,002	W	0,023
K	0,008	X	0,001
L	0,040	Y	0,020
M	0,024	Z	0,001

Uwzględniając obliczone przez siebie prawdopodobieństwa, Beker i Piper dzielią 26 liter na pięć grup:

- (1) E – prawdopodobieństwo wystąpienia ok. 0,120;
- (2) T, A, O, I, N, S, H, R – prawdopodobieństwa wystąpienia od 0,06 do 0,09;
- (3) D, L – każda z prawdopodobieństwem wystąpienia ok. 0,04;
- (4) C, U, M, W, F, G, Y, P, B – z prawdopodobieństwami od 0,015 do 0,028;
- (5) V, K, J, X, Q, Z – każda z prawdopodobieństwem ok. 0,01.

---

<sup>†</sup>Porównaj przypis na str. 4 (przyp. tłum.).

Warto także brać pod uwagę grupy dwóch lub trzech kolejnych liter. Oto 30 najczęściej występujących grup dwuliterowych (w porządku malejącym): *TH, HE, IN, ER, AN, RE, ED, ON, ES, ST, EN, AT, TO, NT, HA, ND, OU, EA, NG, AS, OR, TI, IS, ET, IT, AR, TE, SE, HI* oraz *OF*. Z kolei dwanaście najczęściej występujących grup trzyliterowych (w porządku malejącym) to: *THE, ING, AND, HER, ERE, ENT, THA, NTH, WAS, ETH, FOR* oraz *DTH*.

### 1.2.1. Kryptoanaliza szyfru afanicznego

Zobaczmy na przykładzie **szyfru afanicznego**, jak można dokonywać kryptoanalizy na podstawie danych statystycznych.

#### Przykład 1.9

Przypuśćmy, że Oskar przechwycił następujący tekst zaszyfrowany:

FMXVEDKAPHFERBNDKRXRSREFMORUDSDKDVSHVUFEDKAPRKDLYEVLRHHRH

Tablica 1.2 zawiera dane częstości występowania liter w tym tekście.

**TABLICA 1.2. Częstość występowania 26 liter tekstu zaszyfrowanego**

Litera	Częstość	Litera	Częstość	Litera	Częstość
A	2	J	0	S	3
B	1	K	5	T	2
C	0	L	2	U	2
D	7	M	2	V	4
E	5	N	1	W	0
F	4	O	1	X	2
G	0	P	2	Y	1
H	5	Q	0	Z	0
I	0	R	8		

Tekst zaszyfrowany ma tylko 57 znaków, ale to wystarczy do kryptoanalizy szyfru afanicznego. W tekście najczęściej występują następujące litery: *R* (8 wystąpień), *D* (7 wystąpień), *E, H, K* (5 wystąpień) oraz *F, S, V* (po 4 wystąpienia każda). Możemy przyjąć wstępową hipotezę, że *R* jest odpowiednikiem szyfrowym litery *e*, a *D* reprezentuje literę *t*, gdyż właśnie *e* i *t* są najczęściej występującymi literami. Wyrażając to liczbowo, mamy  $e_K(4) = 17$  oraz  $e_K(19) = 3$ . Przypomnijmy, że  $e_K(x) = ax + b$ , ale wartości *a* i *b* jeszcze nie znamy. Mamy zatem układ dwóch równań liniowych z dwiema niewiadomymi:

$$4a + b = 17$$

$$19a + b = 3.$$

Układ ma tylko jedno rozwiązanie:  $a = 6$ ,  $b = 19$  (w  $\mathbb{Z}_{26}$ ). Niestety, taki klucz nie jest dopuszczalny, ponieważ  $\text{NWD}(a, 26) = 2 > 1$ . Tak więc hipoteza okazała się błędna.

Przypuśćmy teraz, że  $R$  jest odpowiednikiem szyfrowym litery  $e$ , natomiast literę  $t$  reprezentuje  $H$ . Postępując jak wyżej, otrzymujemy  $a = 13$ , a więc znów klucz niedopuszczalny. Próbujmy dalej: niech  $R$  będzie wciąż odpowiednikiem szyfrowym  $e$ , ale tym razem niech  $K$  będzie odpowiednikiem szyfrowym  $t$ . Obliczamy:  $a = 3$ ,  $b = 5$ , co w każdym razie daje klucz spełniający wymagany warunek. Pozostaje obliczyć funkcję deszyfrowania, odpowiadającą kluczowi  $K = (3, 5)$ , a następnie użyć jej do odczytania tekstu zaszyfrowanego, aby sprawdzić, czy wyjdzie z tego sensowny tekst angielski. To potwierdzi trafność wyboru pary  $(3, 5)$ .

Po wykonaniu tych operacji otrzymamy  $d_K(y) = 9y - 19$ , a tekst zaszyfrowany przekształci się w następujący ciąg:

algorithmsarequitegeneraldefinitionsofarithmeticprocesses<sup>†</sup>

Możemy stwierdzić, że ustaliliśmy właściwy klucz. □

### 1.2.2. Kryptoanaliza szyfru podstawieniowego

Przyjrzyjmy się teraz sytuacji bardziej skomplikowanej, czyli szyfrowi podstawieniowemu. Rozpatrzmy następujący tekst zaszyfrowany.

#### Przykład 1.10

Oto tekst zaszyfrowany za pomocą szyfru podstawieniowego:

YIFQFMZRWFYVECFMDZPCVMRZWNMDZVEJBTXCDDUMJ  
NDIFEFDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ  
NZUCDRJXYYSMRTMEYIFZWDYVZVFZUMRZCRWNZDZJJ  
XZWGCHSMRNMDHNCMFQCHZJMZXWIEJYUCFWDJNZDIR

Tablica 1.3 zawiera wyniki analizy częstościowej tego tekstu.

**TABLICA 1.3. Częstość występowania 26 liter tekstu zaszyfrowanego**

Litera	Częstość	Litera	Częstość	Litera	Częstość
A	0	J	11	S	3
B	1	K	1	T	2
C	15	L	0	U	5
D	13	M	16	V	5
E	7	N	9	W	8
F	11	O	0	X	6
G	1	P	1	Y	10
H	4	Q	4	Z	20
I	5	R	10		

<sup>†</sup>W tłumaczeniu: „Algorytmy są całkiem ogólnymi definicjami procesów arytmetycznych” (przyp. tłum.).

Ponieważ litera  $Z$  występuje tu znaczco częściej niż jakikolwiek inny znak tekstu zaszyfrowanego, możemy domniemywać, że  $d_K(Z) = e$ . Ponadto co najmniej 10 razy występują w tekście litery:  $C, D, F, J, M, R, Y$ . Można oczekiwac, że są to odpowiedniki (niektórych spośród) liter  $t, a, o, i, n, s, h, r$ , jednak różnice w częstości występowania są tu zbyt znikome, by można było odkryć powiązania.

W tej sytuacji warto zainteresować się grupami dwuliterowymi, zwłaszcza postaci  $-Z$  lub  $Z-$ , gdyż podejrzewamy, że  $Z$  reprezentuje literę  $e$ . Stwierdzamy, że najczęściej występującymi grupami dwuliterowymi tego rodzaju są  $DZ$  i  $ZW$  (po cztery razy),  $NZ$  i  $ZU$  (po trzy razy) oraz  $RZ, HZ, XZ, FZ, ZR, ZV, ZC, ZD$  i  $ZJ$  (po dwa razy). Ponieważ  $ZW$  występuje cztery razy, a  $WZ$  ani razu, a ponadto  $W$  pojawia się rzadziej niż inne znaki, możemy domniemywać, że  $d_K(W) = d$ . Z kolei z tego, że  $DZ$  występuje cztery razy, a  $ZD$  dwukrotnie, możemy przypuszczać, że  $d_K(D) \in \{r, s, t\}$ , choć nie jest jasne, która z tych trzech możliwości jest poprawna.

Przyjmując założenie, że  $d_K(Z) = e$  oraz  $d_K(W) = d$ , popatrzmy raz jeszcze na tekst zaszyfrowany. Zauważmy, że grupy  $ZRW$  i  $RZW$  występują blisko początku tego tekstu, a grupa  $RW$  powtarza się także w dalszej jego części. Biorąc pod uwagę, że  $R$  pojawia się często w tekście zaszyfrowanym, a grupa  $nd$  jest w języku angielskim często używana, możemy uznać  $d_K(R) = n$  za rozwiązanie najbardziej prawdopodobne.

Znamy teraz następującą odpowiedniość:

```
-----end-----e---ned---e-----  
YIFQFMZRWFQFYVECFMDZPCVMRZNMDZVEJBTXCDDUMJ  
-----e----e-----n=d---en----e----e  
NDIFEFDMDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ  
-----e---n-----n-----ed---e---e---e---ne-nd-e-e--  
NZUCDRJXYYSMRTMEYIFZWDYVZVYFZUMRZCRWNZDZJJ  
-ed-----n-----e---ed-----d---e--n  
XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR
```

Dalej możemy spróbować przyjąć  $d_K(N) = h$ , jako że grupa  $NZ$  występuje często, w odróżnieniu od grupy  $ZN$ . Jeśli mamy rację, to z fragmentu tekstu jawnego  $ne-ndhe$  wynika, że  $d_K(C) = a$ . Uwzględniając te zmiany, mamy:

```
-----end-----a---e-a---nedh---e-----a-----  
YIFQFMZRWFQFYVECFMDZPCVMRZNMDZVEJBTXCDDUMJ  
h-----ea---e-a---a---nhad-a-en--a-e-h--e  
NDIFEFDMDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ  
he-a-n-----n-----ed---e---e---neandhe-e--  
NZUCDRJXYYSMRTMEYIFZWDYVZVYFZUMRZCRWNZDZJJ  
-ed-a---nh---ha---a-e---ed-----a-d--he--n  
XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR
```

Przyjrzyjmy się teraz literze  $M$  sklasyfikowanej na drugim miejscu pod względem częstości występowania w tekście zaszyfrowanym. Fragment  $RNM$  w tym tekście, który uznaliśmy za odpowiednik  $nh-$ , skłania do przypuszczenia, że  $h-$  jest początkiem nowego słowa, zatem  $M$  prawdopodobnie reprezentuje samogłoskę. Uwzględniliśmy już  $a$  i  $e$ , spodziewamy się teraz, że  $d_K(M) = i$  lub  $o$ . W języku angielskim łatwiej o  $ai$  niż o  $ao$ , więc pojawienie się grupy  $CM$  w tekście zaszyfrowanym sugeruje, że lepiej zacząć od  $d_K(M) = i$ . Otrzymujemy wówczas:

```
-----iend-----a-i-e-a-inedhi-e-----a---i-
YIFQFMZRWFYVECFMDZPCVMRZNMDZVEJBTXCDDUMJ

h-----i-ea-i-e-a---a-i-nhad-a-en--a-e-hi-e
NDIFEFDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ

he-a-n----in-i----ed---e---e-inelandhe-e--
NZUCDRJXXYSMRTMEYIFZWDYVZVFZUMRZCRWNZDZJJ

-ed-a--inhi--hai--a-e-i--ed----a-d--he--n
XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR
```

Teraz możemy spróbować ustalić literę, która jest odpowiednikiem szyfrowym  $o$ . Ponieważ ta samogłoska występuje dość często, więc jej szyfrowym odpowiednikiem mogłaby być jedna z liter  $D, F, J, Y$ . Wybór  $D, F$  lub  $J$  spowodowałby pojawienie się w tekście jawnym długich ciągów samogłosek, mianowicie ciągu  $aoi$ , odpowiadającego  $CFM$  lub  $CJM$ , zatem najbardziej prawdopodobnym odpowiednikiem szyfrowym litery  $o$  jest  $Y$ . Przyjmijmy więc  $d_K(Y) = o$ .

Wśród nieroznanych jeszcze liter tekstu zaszyfrowanego  $D, F$  i  $J$  występują najczęściej, mogą więc odpowiadać w pewnej kolejności literom  $r, s$  i  $t$ . Dwukrotne wystąpienie grupy trzyliterowej  $NMD$  może wskazywać na to, że  $d_K(D) = s$ , co daje w tekście jawnym grupę *his* (*jego*) (i pozostaje zgodne z naszą wcześniejszą hipotezą, że  $d_K(D) \in \{r, s, t\}$ ). Fragment  $HNCMF$  mógłby być szyfrem słowa *chair* (*krzesło*), z czego wynikałoby, że  $d_K(F) = r$  (oraz  $d_K(H) = c$ ) i w wyniku eliminacji także  $d_K(J) = t$ . Otrzymamy wówczas:

```
o-r-riend-ro--arise-a-inedhise--t---ass-it
YIFQFMZRWFYVECFMDZPCVMRZNMDZVEJBTXCDDUMJ

hs-r-riseasi-e-a-orationhadta-en--ace-hi-e
NDIFEFDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ

he-asnt-oo-in-i-o-redso-e-ore--ineandhesett
NZUCDRJXXYSMRTMEYIFZWDYVZVFZUMRZCRWNZDZJJ

-ed-ac-inhischair-aceti-ted--to-ardsthes-n
XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR
```

Ustalenie tekstu jawnego oraz klucza w przykładzie 1.10 nie sprawia już teraz trudności. Po deszyfrowaniu otrzymujemy następujący tekst:

Our friend from Paris examined his empty glass with surprise, as if evaporation had taken place while he wasn't looking. I poured some more wine and he settled back in his chair, face tilted towards the sun<sup>1</sup>.

□

### 1.2.3. Kryptoanaliza szyfru Vigenère'a

Zajmiemy się w tym podrozdziale niektórymi metodami kryptoanalizy szyfru Vigenère'a. Pierwsza rzecz to rozpoznanie długości słowa kluczowego; oznaczmy ją przez  $m$ . Możemy tu zastosować kilka różnych metod. Pierwszą z nich jest tzw. *test Kasiskiego*, w drugiej wykorzystuje się *indeks zgodności* (ang. *index of coincidence*).

Test Kasiskiego wziął nazwę od nazwiska Friedricha Kasiskiego, który opisał go po raz pierwszy w 1863 roku. Opiera się na obserwacji, że dwa identyczne fragmenty tekstu jawnego zostaną identycznie zaszyfrowane, jeśli w tekście jawnym jeden z nich jest przesunięty względem drugiego o  $x$  pozycji, przy czym  $x \equiv 0 \pmod{m}$ . Odwrotnie, jeśli w tekście zaszyfrowanym znajdujemy dwa identyczne fragmenty o długości co najmniej, powiedzmy, trzech znaków, to istnieje niemałe prawdopodobieństwo, że odpowiadają one identycznym fragmentom tekstu jawnego.

Test ten działa następująco. W tekście zaszyfrowanym szukamy par identycznych fragmentów o długości co najmniej trzech znaków, zapamiętując odległości między pierwszymi znakami fragmentów należących do jednej pary. Mając takie odległości  $d_1, d_2, \dots$ , zakładamy, że  $m$  dzieli największy wspólny dzielnik tych liczb.

Dalsze informacje dotyczące wartości  $m$  można uzyskać za pomocą indeksu zgodności. Pojęcie to zdefiniował w 1920 roku Wolfe Friedman<sup>†</sup> w sposób następujący:

#### DEFINICJA 1.7

Niech  $\mathbf{x} = x_1x_2\dots x_n$  będzie ciągiem  $n$  znaków alfabetu. *Indeksem zgodności*, oznaczanym symbolem  $I_c(\mathbf{x})$ , nazywamy prawdopodobieństwo tego, że dwa losowo wybrane wyrazy ciągu  $\mathbf{x}$  są identyczne. Niech częstotliwości wystąpienia

<sup>1</sup>P. Mayle: *A Year in Provence*, A. Knopf, Inc., 1989. [W tłumaczeniu: Nasz przyjaciel z Paryża obejrzał swój pusty kieliszek ze zdziwieniem, jakby zawartość wyparowała, gdy nań nie patrzył. Nalałem mu wina i ponownie oparł się wygodnie, z twarzą zwróconą do słońca (przyp. tłum.).]

<sup>†</sup>Słynny kryptolog amerykański znany pod imieniem William, które nadali mu rodzice po wyemigrowaniu do Stanów Zjednoczonych (przyp. tłum.).

$A, B, C, \dots, Z$  w  $\mathbf{x}$  będą odpowiednio równe  $f_0, f_1, \dots$ . Dwa wyrazy z ciągu  $\mathbf{x}$  można wybrać na  $\binom{n}{2}$  sposobów<sup>2</sup>. Dla każdego  $i$ ,  $0 \leq i \leq 25$ , mamy  $\binom{f_i}{2}$  sposobów wybrania dwóch elementów  $i$ . Stąd wzór:

$$I_c(\mathbf{x}) = \frac{\sum_{i=0}^{25} f_i(f_i - 1)}{n(n-1)}.$$

Założymy teraz, że  $\mathbf{x}$  jest ciągiem znaków tekstu w języku angielskim. Oznaczmy oczekiwane prawdopodobieństwa wystąpienia liter  $A, B, \dots, Z$  z tabeli 1.1 odpowiednio przez  $p_0, \dots, p_{25}$ . Wówczas

$$I_c(\mathbf{x}) \approx \sum_{i=0}^{25} p_i^2 = 0,065,$$

ponieważ prawdopodobieństwo, że dwa razy losowo wybierzemy literę  $A$  jest równe  $p_0^2$ , prawdopodobieństwo dwukrotnego wyboru litery  $B$  jest równe  $p_1^2$  itd. To samo rozumowanie odnosi się do ciągu  $\mathbf{x}$  uzyskanego w wyniku zastosowania dowolnego szyfru monoalfabetycznego.

W tym przypadku poszczególne prawdopodobieństwa mogą podlegać permutacji, ale ogólny wynik, czyli

$$\sum_{i=0}^{25} p_i^2,$$



pozostaje niezmienny.

Przypuśćmy, że zaczynamy od tekstu zaszyfrowanego  $\mathbf{y} = y_1 y_2 \dots y_n$ , powstały w wyniku zastosowania szyfru Vigenère'a. Utwórzmy  $m$  podciągów  $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_m$  ciągu  $\mathbf{y}$ , zapisując tekst zaszyfrowany w kolumnach w prostokątnej macierzy wymiaru  $m \times (n/m)$ . Wierszami w tej macierzy są podciagi  $\mathbf{y}_i$ ,  $1 \leq i \leq m$ . Jeśli  $m$  jest rzeczywiście długością klucza, to każda z liczb  $I_c(\mathbf{y}_i)$  powinna być w przybliżeniu równa 0,065. Jednak, jeśli długość klucza nie jest równa  $m$ , to podciagi  $\mathbf{y}_i$  będą miały znacznie bardziej losowy charakter, ponieważ powstały z zastosowania różnych kluczy. Zauważmy, że dla całkowicie losowo wybranego ciągu

$$I_c \approx 26(1/26)^2 = 1/26 = 0,038.$$

Te dwie wartości – 0,065 i 0,038 – są wystarczająco odległe, by w wielu przypadkach umożliwić ustalenie poprawnej długości klucza (lub potwierdzenie hipotezy powiększej po zastosowaniu testu Kasiskiego).

Zilustrujmy obie te metody przykładem.

---

<sup>2</sup> Współczynnik dwumianowy albo symbol Newtona  $\binom{n}{k} = \frac{n!}{k!(n-k)!}$  określa liczbę sposobów wybrania  $k$ -elementowego podzbioru ze zbioru  $n$ -elementowego.

### Przykład 1.11

Oto tekst otrzymany po zastosowaniu szyfru Vigenère'a:

```
CHREEVOAHMAERATBIAXXWTNXBEEOPHBSBQMQEQRBW
RVXUOAKXAOSXXWEAHBWGJMMQMNKGKGRFVGWTRZXWIAK
LXFPSKAUTEMNDCMGTSXMXBTUIADNGMGPSRELXNJELX
VRVPRTULHDNQWTWDTYGBPHXTFALJHASVBFXNGLLCHR
ZBWELEKMSJIKNBHWRJGNMGJSLXFYYPHAGNRBIEQJT
AMRVLCCRREMNDGLXRRIMGNSNRWCHRQHAEYEVTAQEBBI
PEEVVKAKOEWADREMXTBHHCHRTKDNRZCHRCLQQHP
WQAIIXWXNRMGWOIIFKEE
```

Zaczniemy od testu Kasiskiego. Ciąg *CHR* pojawia się w pięciu miejscach tekstu zaszyfrowanego, z pierwszą literą na pozycjach: 1, 166, 236, 276 i 286. Odległości od jego pierwszego wystąpienia do pozostałych czterech są równe (odpowiednio) 165, 235, 275 i 285. Największym wspólnym dzielnikiem tych liczb jest 5 i jest bardzo prawdopodobne, że to jest właśnie długość klucza.

Sprawdźmy, czy obliczenie indeksów zgodności prowadzi do tego samego wniosku. Gdy  $m = 1$ , indeks zgodności jest równy 0,045. Dla  $m = 2$  indeksy są równe 0,046 i 0,041. Dla  $m = 3$  mamy 0,043, 0,050 oraz 0,047. Gdy  $m = 4$ , indeksy są równe 0,042, 0,039, 0,046, 0,040. Wreszcie dla  $m = 5$  otrzymujemy wartości: 0,063, 0,068, 0,069, 0,061 oraz 0,072. To także daje mocne podstawy do przypuszczenia, że długość klucza wynosi 5.  $\square$

Jak na podstawie takiego założenia określić klucz? Warto tu odwołać się do pojęcia wzajemnego indeksu zgodności (ang. *mutual index of coincidence*) dwóch ciągów.

### DEFINICJA 1.8

Niech  $\mathbf{x} = x_1x_2\dots x_n$  i  $\mathbf{y} = y_1y_2\dots y_{n'}$  będą ciągami złożonymi z  $n$  i  $n'$  znaków, odpowiednio. *Wzajemny indeks zgodności*  $\mathbf{x}$  i  $\mathbf{y}$ , oznaczany symbolem  $MI_c(\mathbf{x}, \mathbf{y})$ , określamy jako prawdopodobieństwo tego, że losowo wybrany element ciągu  $\mathbf{x}$  będzie równy losowo wybranemu elementowi ciągu  $\mathbf{y}$ . Jeśli częstości występowania elementów  $A, B, C, \dots, Z$  w  $\mathbf{x}$  i  $\mathbf{y}$  oznaczymy odpowiednio przez  $f_0, f_1, \dots, f_{25}$  i  $f'_0, f'_1, \dots, f'_{25}$ , to  $MI_c(\mathbf{x}, \mathbf{y})$  można opisać następującym wzorem:

$$MI_c(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=0}^{25} f_i f'_i}{nn'}.$$

Teraz, przyjmując, że znamy już wartość  $m$ , podciągi  $\mathbf{y}_i$  powstają z szyfrowania tekstu jawnego z przesunięciem. Niech  $K = (k_1, k_2, \dots, k_m)$  będzie kluczem. Spróbujmy oszacować wartość  $MI_c(\mathbf{y}_i, \mathbf{y}_j)$ . Przypuśćmy, że wybieramy losowo jeden znak w  $\mathbf{y}_i$  oraz jeden znak w  $\mathbf{y}_j$ . Prawdopodobieństwo otrzymania dwa razy  $A$  jest równe  $p_{-k_i} p_{-k_j}$ , otrzymania dwa razy  $B$  to  $p_{1-k_i} p_{1-k_j}$  itd. (Zauważmy

że wszystkie indeksy dolne są brane modulo 26). Dochodzimy zatem do takiego oszacowania:

$$MI_c(\mathbf{y}_i, \mathbf{y}_j) \approx \sum_{h=0}^{25} p_{h-k_i} p_{h-k_j} = \sum_{h=0}^{25} p_h p_{h+k_i-k_j}.$$

Wartość tego oszacowania zależy tylko od różnicy  $k_i - k_j \bmod 26$ , którą nazywamy *względnym przesunięciem*  $\mathbf{y}_i$  i  $\mathbf{y}_j$ . Zauważmy ponadto, że

$$\sum_{h=0}^{25} p_h p_{h+l} = \sum_{h=0}^{25} p_h p_{h-l},$$

a więc względne przesunięcie o  $l$  daje to samo oszacowanie wartości  $MI_c$ , co względne przesunięcie o  $26 - l$ .

Tablica 1.4 zawiera oszacowanie wartości odpowiadające względnym przesunięciom w zakresie od 0 do 13.

**TABLICA 1.4. Oczekiwane wzajemne indeksy zgodności**

Względne przesunięcie	Wartość oczekiwana $MI_c$
0	0,065
1	0,039
2	0,032
3	0,034
4	0,044
5	0,033
6	0,036
7	0,039
8	0,034
9	0,034
10	0,038
11	0,045
12	0,039
13	0,043

Warto tu przede wszystkim zauważyć, że jeśli względne przesunięcie nie jest zerowe, to oszacowania wahają się między 0,031 i 0,045, podczas gdy zerowe względne przesunięcie daje oszacowanie 0,065. Tę obserwację możemy wykorzystać do sformułowania prawdopodobnej hipotezy dotyczącej wartości  $l = k_i - k_j$ , względnego przesunięcia  $\mathbf{y}_i$  i  $\mathbf{y}_j$ , w sposób następujący. Przypuśćmy, że ustalamy  $\mathbf{y}_i$  i rozpatrujemy wynik zaszyfrowania  $\mathbf{y}_j$  za pomocą ciągu  $e_0, e_1, e_2, \dots$ . Oznaczmy otrzymane w ten sposób ciągi przez  $\mathbf{y}_j^0, \mathbf{y}_j^1$  itd. Obliczenie indeksów  $MI_c(\mathbf{y}_i, \mathbf{y}_j^g)$ , gdy  $0 \leq g \leq 25$ , nie sprawia trudności. Można to zrobić, stosując wzór:

$$MI_c(\mathbf{x}, \mathbf{y}^g) = \frac{\sum_{i=0}^{25} f_i f'_{i-g}}{nn'}.$$

Gdy  $g = l$ , wartość  $MI_c$  powinna być bliska 0,065, ponieważ względne przesunięcie  $\mathbf{y}_i$  i  $\mathbf{y}_j$  jest równe zeru. Natomiast gdy  $g \neq l$ , powinna ona utrzymywać się w granicach od 0,031 do 0,045.

Stosując taką metodę, możemy otrzymać względne przesunięcia dla dowolnych dwóch podciągów  $\mathbf{y}_i$ . Zostajemy wtedy z 26 możliwymi kluczami, które łatwo już znaleźć, na przykład, metodą wyczerpującego przeszukiwania kluczy.

Aby to zilustrować, powróćmy do przykładu 1.11.

### Przykład 1.11 cd.

Przyjęliśmy hipotezę, że klucz ma długość 5. Spróbujmy teraz obliczyć względne przesunięcia. Używając komputera, nietrudno obliczyć 260 wartości  $MI_c(\mathbf{y}_i, \mathbf{y}_j^g)$ , gdy  $1 \leq i < j \leq 5$ ,  $0 \leq g \leq 25$ . Są one zamieszczone w tablicy 1.5. Dla każdej pary  $(i, j)$  szukamy wartości  $MI_c(\mathbf{y}_i, \mathbf{y}_j^g)$  bliskich 0,065. Jeśli istnieje tylko jedna taka wartość (dla danej pary  $(i, j)$ ), przyjmujemy domniemanie, że to jest właśnie wartość względnego przesunięcia.

W tablicy 1.5 sześć wartości wpisano w ramki. Stanowią one silne poszukiwające na to, że: względne przesunięcie  $\mathbf{y}_1$  i  $\mathbf{y}_2$  jest równe 9, względne przesunięcie  $\mathbf{y}_1$  i  $\mathbf{y}_5$  jest równe 16, względne przesunięcie  $\mathbf{y}_2$  i  $\mathbf{y}_3$  jest równe 13, względne przesunięcie  $\mathbf{y}_2$  i  $\mathbf{y}_5$  jest równe 7, względne przesunięcie  $\mathbf{y}_3$  i  $\mathbf{y}_5$  jest równe 20, a względne przesunięcie  $\mathbf{y}_4$  i  $\mathbf{y}_5$  jest równe 11. Daje to następujący układ równań z pięcioma niewiadomymi  $k_1, k_2, k_3, k_4, k_5$ :

$$k_1 - k_2 = 9$$

$$k_1 - k_5 = 16$$

$$k_2 - k_3 = 13$$

$$k_2 - k_5 = 7$$

$$k_3 - k_5 = 20$$

$$k_4 - k_5 = 11.$$

Pozwala to wyrazić pięć niewiadomych za pomocą  $k_1$ :

$$k_2 = k_1 + 17$$

$$k_3 = k_1 + 4$$

$$k_4 = k_1 + 21$$

$$k_5 = k_1 + 10.$$

Tak więc jest bardzo prawdopodobne, że klucz ma postać  $(k_1, k_1 + 17, k_1 + 4, k_1 + 21, k_1 + 10)$  dla pewnej wartości  $k_1 \in \mathbb{Z}_{26}$ . Możemy teraz podejrzewać, że klucz jest pewnym cyklicznym przesunięciem *AREV K*. Szybko dochodzimy do wniosku, że słowem kluczowym jest *JANET*. Całkowicie rozszыfrowany tekst wygląda tak:

TABLICA 1.5. Rzeczywiste wzajemne indeksy zgodności

$i$	$j$	Wartości $MI_c(\mathbf{y}_i \mathbf{y}_j^g)$								
1	2	0,028	0,027	0,028	0,034	0,039	0,037	0,026	0,025	0,052
		0,068	0,044	0,026	0,037	0,043	0,037	0,043	0,037	0,028
		0,041	0,041	0,034	0,037	0,051	0,045	0,042	0,036	
1	3	0,039	0,033	0,040	0,034	0,028	0,053	0,048	0,033	0,029
		0,056	0,050	0,045	0,039	0,040	0,036	0,037	0,032	0,027
		0,037	0,036	0,031	0,037	0,055	0,029	0,024	0,037	
1	4	0,034	0,043	0,025	0,027	0,038	0,049	0,040	0,032	0,029
		0,034	0,039	0,044	0,044	0,034	0,039	0,045	0,044	0,037
		0,055	0,047	0,032	0,027	0,039	0,037	0,039	0,035	
1	5	0,043	0,033	0,028	0,046	0,043	0,044	0,039	0,031	0,026
		0,030	0,036	0,040	0,041	0,024	0,019	0,048	0,070	0,044
		0,028	0,038	0,044	0,043	0,047	0,033	0,026	0,046	
2	3	0,046	0,048	0,041	0,032	0,036	0,035	0,036	0,030	0,024
		0,039	0,034	0,029	0,040	0,067	0,041	0,033	0,037	0,045
		0,033	0,033	0,027	0,033	0,045	0,052	0,042	0,030	
2	4	0,046	0,034	0,043	0,044	0,034	0,031	0,040	0,045	0,040
		0,048	0,044	0,033	0,024	0,028	0,042	0,039	0,026	0,034
		0,050	0,035	0,032	0,040	0,056	0,043	0,028	0,028	
2	5	0,033	0,033	0,036	0,046	0,026	0,018	0,043	0,080	0,050
		0,029	0,031	0,045	0,039	0,037	0,027	0,026	0,031	0,039
		0,040	0,037	0,041	0,046	0,045	0,043	0,035	0,030	
3	4	0,038	0,036	0,040	0,033	0,036	0,060	0,035	0,041	0,029
		0,058	0,035	0,035	0,034	0,053	0,030	0,032	0,035	0,036
		0,036	0,028	0,046	0,032	0,051	0,032	0,034	0,030	
3	5	0,035	0,034	0,034	0,036	0,030	0,043	0,043	0,050	0,025
		0,041	0,051	0,050	0,035	0,032	0,033	0,033	0,052	0,031
		0,027	0,030	0,072	0,035	0,034	0,032	0,043	0,027	
4	5	0,052	0,038	0,033	0,038	0,041	0,043	0,037	0,048	0,028
		0,028	0,036	0,061	0,033	0,033	0,032	0,052	0,034	0,027
		0,039	0,043	0,033	0,027	0,030	0,039	0,048	0,035	

The almond tree was in tentative blossom. The days were longer, often ending with magnificent evenings of corrugated pink skies. The hunting season was over, with hounds and guns put away for six months. The vineyards were busy again as the well-organized farmers treated their vines and the more lackadaisical neighbors hurried to do the pruning they should have done in November<sup>3</sup>. □

<sup>3</sup>P. Mayle, *A Year in Provence*, A. Knopf, Inc., 1989. [W tłumaczeniu: Drzewo migdałowe zaczynało nieśmiało rozwijać się. Dzień trwał dłużej, nierzadko uwieńczony olśniewającym wieczorem pod pośladkowanym, różowym niebem. Skończył się sezon łowiecki, charty i strzelby odstawiono na sześć miesięcy. W winnicach ponownie zapanował ruch, gdy lepiej zorganizowani gospodarze obrabiali swoje winorośla, a ich mniej zapobiegliwi sąsiedzi rzucali się do przycinania krzewów, które powinni byli przyciąć w listopadzie (przyp. tłum.)].

### 1.2.4. Atak ze znanym tekstem jawnym na szyfr Hilla

Szyfr Hilla jest trudniejszy do złamania z wykorzystaniem tylko tekstu zaszyfrowanego, ale łatwo poddaje się atakowi opartemu na znanym tekście jawnym. Założymy najpierw, że przeciwnik ustalił już zastosowaną wartość  $m$ . Przypuśćmy dalej, że dysponuje on co najmniej  $m$  różnymi parami ciągów  $m$ -wyrazowych,  $x_j = (x_{1,j}, x_{2,j}, \dots, x_{m,j})$  oraz  $y_j = (y_{1,j}, y_{2,j}, \dots, y_{m,j})$  ( $1 \leq j \leq m$ ), takimi że  $y_j = e_K(x_j)$ ,  $1 \leq j \leq m$ . Zapisując wyrazy tych ciągów w postaci dwóch macierzy  $m \times m$ :  $X = (x_{i,j})$  i  $Y = (y_{i,j})$ , mamy równanie macierzowe  $Y = XK$ , gdzie  $K$ , również macierz  $m \times m$ , jest nieznanym jeszcze kluczem. Jeśli tylko macierz  $K$  jest odwracalna, Oskar może obliczyć  $K = X^{-1}Y$ , łamiąc w ten sposób system. (Jeśli  $X$  nie jest odwracalna, trzeba ponowić atak z innymi próbками  $m$  par tekstów jawnych i zaszyfrowanych).

Popatrzmy na prosty przykład.

#### Przykład 1.12

Przypuśćmy, że zastosowano szyfr Hilla z  $m = 2$  do utajnienia tekstu jawnego *friday*, otrzymując tekst zaszyfrowany *PQCFKU*.

Mamy zatem:  $e_K(5, 17) = (15, 16)$ ,  $e_K(8, 3) = (2, 5)$  oraz  $e_K(0, 24) = (10, 20)$ . Z pierwszych dwóch par (tekst jawnny – tekst zaszyfrowany) otrzymujemy równanie macierzowe:

$$\begin{pmatrix} 15 & 16 \\ 2 & 5 \end{pmatrix} = \begin{pmatrix} 5 & 17 \\ 8 & 3 \end{pmatrix} K.$$

Dzięki twierdzeniu 1.3 łatwo obliczamy:

$$\begin{pmatrix} 5 & 17 \\ 8 & 3 \end{pmatrix}^{-1} = \begin{pmatrix} 9 & 1 \\ 2 & 15 \end{pmatrix},$$

zatem

$$K = \begin{pmatrix} 9 & 1 \\ 2 & 15 \end{pmatrix} \begin{pmatrix} 15 & 16 \\ 2 & 5 \end{pmatrix} = \begin{pmatrix} 7 & 19 \\ 8 & 3 \end{pmatrix}.$$

Wynik można sprawdzić na trzeciej parze tekstów, jawnego i zaszyfrowanego.  $\square$

Co może zrobić przeciwnik, gdy nie zna wartości  $m$ ? Zakładając, że nie jest to zbyt duża liczba, może po prostu wypróbować kolejne wartości  $m = 2, 3, \dots$ , dopóki nie natrafi na klucz. Jeśli wybierze wartość niepoprawną, to macierz  $m \times m$  utworzona w wyniku zastosowania opisanego wyżej algorytmu będzie dawała niezgodne wyniki dla pozostałych par tekstów. W ten sposób można ustalić wartość  $m$ , jeśli nie była wcześniej znana.

### 1.2.5. Kryptoanaliza szyfru strumieniowego opartego na LFSR

Przypomnijmy, że tekst zaszyfrowany powstaje jako różnica symetryczna tekstu jawnego i klucza strumieniowego, czyli  $y_i = x_i + z_i \bmod 2$ . Klucz strumieniowy powstaje z  $z_1, \dots, z_m$  na podstawie liniowej zależności rekurencyjnej:

$$z_{m+i} = \sum_{j=0}^{m-1} c_j z_{i+j} \bmod 2,$$

gdzie  $c_0, \dots, c_{m-1} \in \mathbb{Z}_2$  (oraz  $c_0 = 1$ ).

Ponieważ wszystkie operacje w tym kryptosystemie są liniowe, nasuwa się podejrzenie, że, podobnie jak w przypadku szyfru Hilla, system jest podatny na ataki oparte na znanym tekście jawnym. Przypuśćmy, że Oskar zna tekst jawny  $x_1 x_2 \dots x_n$  oraz odpowiadający mu ciąg tekstu zaszyfrowanego  $y_1 y_2 \dots y_n$ . Może wówczas obliczyć bity klucza strumieniowego:  $z_i = x_i + y_i \bmod 2$ ,  $1 \leq i \leq n$ . Założymy także, że Oskar zna wartość  $m$ . Wówczas do odtworzenia całego klucza strumieniowego pozostaje mu tylko obliczyć  $c_0, \dots, c_{m-1}$ . Inaczej mówiąc, musi umieć określić wartości  $m$  niewiadomych.

Dla dowolnego  $i \geq 1$  mamy

$$z_{m+i} = \sum_{j=0}^{m-1} c_j z_{i+j} \bmod 2,$$

a więc równanie liniowe z  $m$  niewiadomymi. Gdy  $n \geq 2m$ , otrzymujemy układ  $m$  równań liniowych z  $m$  niewiadomymi, który można rozwiązać.

Układ  $m$  równań liniowych możemy zapisać w postaci macierzowej:

$$(z_{m+1}, z_{m+2}, \dots, z_{2m}) = (c_0, c_1, \dots, c_{m-1}) \begin{pmatrix} z_1 & z_2 & \dots & z_m \\ z_2 & z_3 & \dots & z_{m+1} \\ \vdots & \vdots & & \vdots \\ z_m & z_{m+1} & \dots & z_{2m-1} \end{pmatrix}.$$

Jeśli macierz współczynników jest odwracalna (modulo 2), uzyskujemy rozwiązanie:

$$(c_0, c_1, \dots, c_{m-1}) = (z_{m+1}, z_{m+2}, \dots, z_{2m}) \begin{pmatrix} z_1 & z_2 & \dots & z_m \\ z_2 & z_3 & \dots & z_{m+1} \\ \vdots & \vdots & & \vdots \\ z_m & z_{m+1} & \dots & z_{2m-1} \end{pmatrix}^{-1}.$$

Zauważmy, że macierz będzie odwracalna, gdy  $m$  jest stopniem rekurencji użytej do generowania klucza strumieniowego (dowód tego faktu znajduje się w ćwiczeniach).

I znów popatrzymy na przykład.

**Przykład 1.13**

Załóżmy, że Oskar zdobywa ciąg tekstu zaszyfrowanego

101101011110010

odpowiadający tekstowi jawnemu

011001111111000.

Może wówczas obliczyć bity klucza strumieniowego:

110100100001010.

Przypuśćmy także, że Oskar wie, iż klucz strumieniowy został wygenerowany za pomocą 5-bitowego LFSR. Rozwiąże wtedy następujące równanie macierzowe otrzymane z pierwszych 10 bitów klucza strumieniowego:

$$(0, 1, 0, 0, 0) = (c_0, c_1, c_2, c_3, c_4) \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

Można sprawdzić, że

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}^{-1} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

Mamy zatem następujący wynik:

$$(c_0, c_1, c_2, c_3, c_4) = (0, 1, 0, 0, 0) \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix} = (1, 0, 0, 1, 0).$$

Rekurencja użyta do wygenerowania klucza strumieniowego ma następującą postać:

$$z_{i+5} = z_i + z_{i+3} \bmod 2.$$

□

### 1.3. Uwagi

Wiele informacji o klasycznej kryptografii można znaleźć w podręcznikach, na przykład: Beker i Piper [BP82] oraz Denning [DE92]. Oszacowania prawdopodobieństw dla 26 znaków alfabetu pochodzą z książki Bekera i Pipera, podobnie jak zmodyfikowana tu kryptoanaliza szyfru Vigenère'a.

Rosen [Ro93] jest dobrym źródłem wiedzy o elementarnej teorii liczb, natomiast Anton [AN91] stanowi wprowadzenie do pojęć elementarnej algebry liniowej.

Książka „Łamacze kodów” Kahna [KA67] jest zajmującą i pouczającą historią kryptografii do 1967 roku. Autor stwierdza w niej, że szyfr Vigenère'a jest przypisywany Vigenère'owi niesłusznie.

Pierwszy opis szyfru Hilla pojawił się w [Hi29], natomiast o szyfrach strumieniowych można się wiele dowiedzieć z książki Rueppela [RU86].

## Ćwiczenia

- 1.1. Niżej znajdują się cztery przykładowe teksty zaszyfrowane. Jeden z nich otrzymało za pomocą szyfru podstawieniowego, do innego został użyty szyfr Vigenère'a, kolejny tekst pochodzi z szyfru afincznego, o jednym z tekstów nie wiemy, w jaki sposób powstał. W każdym przypadku należy ustalić tekst jawnny.

Opisz czytelnie kroki podjęte w celu rozszyfrowania każdego z tych tekstów. W opisie powinny mieścić się wszystkie wykonane analizy statystyczne i obliczenia.

Pierwsze dwa teksty jawnie pochodzą z „The Diary of Samuel Marchbanks” Robertsona Daviesa, Clarke Irwin, 1947; czwarty jest fragmentem „Lake Wobegon Days” Garrisona Keillora, Viking Penguin, Inc., 1985.

- (a) szyfr podstawieniowy:

```
EMGLOSUDCGDNCUSWYSFHNSFCYKDPUMLWGYICOYXSIJCK
QPKUGKMGOLICGINCGACKSNISACYKZSCKXECJCKSHYSXCG
OIDPKZCNKSHICGIWYGGKGKGOLDSILKGIOUSIGLEDSPWZU
GFZCNDGYYSFUSZCNXEJNCGYEOWEUPXEZGACGNFGLKNS
ACIGOIYCKXCJUCIUZCFZCCNDGYYSFUEUKUZCSOCPZCCNC
IACZEJNC SHFZEJZEGMXCYHCJUMGKUCY
```

**WSKAZÓWKA**  $F$  jest kodem litery  $w$ .

- (b) szyfr Vigenère'a:

```
KCCPKBGUFDPHQTYAVINRRTMVGRKDNBVFDETDGILTDXRGUD
DKOTFMBPVGEGLTGCKQRACQCDNAWC RXIZAKFTLEWRPTYC
QKYVXCHKFTPONCQQRHJV AJUWETMCMSPKQDYHJVDAHCTRL
SVSKCGCZQQDZXGSFRLSWCWSJTBAFSIASPRJAHKJRJUMV
GKMITZHFPDISPZLVLGWTPLKKEBDPGCEBSHTJRWXBAFS
PEZQNRWXCVYCGAONWDDKACKAWBBIKFTIOVKCGGHJVLNHI
FFSQESVYCLACNVRWBIREPBVFEXOSCDYGYZWPFDTKFQIY
CWHJVLNHIQIBTKHJVNPIST
```

(c) szyfr afiniczny:

KQEREJEBCPJCJRKEACUZBKRVPKRBCIBQCABJCVFCUP  
 KRIOKPACUZQEPBKRXPEIIEABDKPBCPFCDCCAFIEABDKP  
 BCPFEQPKAZBKRHAIBKAPCCIBURCCDKDCCJCIDFUIXPRAFF  
 ERBICZDFKABICBBENEFCUPJCVKABPCYDCCDPKBCCPERK  
 IVKSCPICBRKIJPKABI

(d) nieznany szyft:

BVVSNSIHQCEELSSKKYERIF JKKXUMBGYKAMQL JTYAVFBKVT  
 DVBPVVRJYYLAOKYMPQSCGDLFSRLLPROYGESEBUALRWXM  
 MASAZLGLEDF JBZAVPVXWICGJXASCBYEHOSNMULKEAHTQ  
 OKMFLEBKFXLRRFDTZXCIBJSICCBGAWDVYDHAVFJXZIBKC  
 GJIWEAHTTOEWTHKRQVVRGZBXVIREMMASCSPBNLHJMBLR  
 FFJELHWETYWLWISTFVVVFJCMHYUYRUFSPMGESIGRLWALSWM  
 NUHSIMYYITCCQPSZSICEHCCMZFEVGJVYUCDEMMPGHVAAUM  
 ELCMOEHVLTIPSYUILVGFMLVWDVYDBTHFRAYISYSKGVSUU  
 HYHGGCKTMBLRX

1.2. (a) Ile jest odwracalnych macierzy  $2 \times 2$  nad  $\mathbb{Z}_{26}$ ?

(b) Niech  $p$  będzie liczbą pierwszą. Wykaż, że liczba odwracalnych macierzy  $2 \times 2$  nad  $\mathbb{Z}_p$  jest równa  $(p^2 - 1)(p^2 - p)$ .

**WSKAZÓWKA**  $\mathbb{Z}_p$  jest ciałem, gdyż  $p$  jest liczbą pierwszą. Wykorzystaj fakt, że macierz nad ciałem jest odwracalna wtedy i tylko wtedy, gdy jej wiersze są wektorami liniowo niezależnymi (co oznacza, że nie istnieje niezerowa kombinacja liniowa wierszy, której sumą byłby wektor zerowy).

(c) Znajdź ogólny wzór na liczbę odwracalnych macierzy  $m \times m$  nad  $\mathbb{Z}_p$ , gdy  $p$  jest liczbą pierwszą oraz  $m \geq 2$ .

1.3. Niekiedy wygodnie jest tak dobrać klucz, aby operacja deszyfrowania była tożsama z operacją szyfrowania. W przypadku szyfru Hilla szukalibyśmy takiej macierzy  $K$ , że  $K = K^{-1}$  (taką macierz nazywa się macierzą *involutywną*). Hill rzeczywiście zalecał stosowanie w jego szyfrze kluczy w postaci macierzy involutywnych. Określ liczbę macierzy involutywnych (nad  $\mathbb{Z}_{26}$ ) w przypadku  $m = 2$ .

**WSKAZÓWKA** Użyj wzoru z twierdzenia 1.3 i zauważ, że  $\det A = \pm 1$  dla dowolnej macierzy involutywnej  $A$  nad  $\mathbb{Z}_{26}$ .

1.4. Wyznacz macierz szyfrowania, jeśli wiadomo, że tekst jawnny

breathtaking

po zaszyfrowaniu szyfrem Hilla (z nieznaną wartością  $m$ ) daje następujący ciąg:

RUPOTENTOSUP

1.5. **Afinicznym szyfrem Hilla** nazywamy następującą modyfikację szyfru Hilla: Niech  $m$  będzie liczbą całkowitą dodatnią i niech  $\mathcal{P} = \mathcal{C} = (\mathbb{Z}_{26})^m$ . W tym systemie kryptograficznym kluczem  $K$  jest para  $(L, b)$ , w której  $L$  jest odwracalną macierzą  $m \times m$  nad  $\mathbb{Z}_{26}$ , a  $b \in (\mathbb{Z}_{26})^m$ . Dla  $x = (x_1, \dots, x_m) \in \mathcal{P}$  i  $K = (L, b) \in \mathcal{K}$  obliczamy  $y = e_K(x) = (y_1, \dots, y_m)$  ze wzoru  $y = xL + b$ . Stąd, jeśli  $L = (l_{i,j})$  i  $b = b_1, \dots, b_m$ , to

$$(y_1, \dots, y_m) = (x_1, \dots, x_m) \begin{pmatrix} l_{1,1} & l_{1,2} & \dots & l_{1,m} \\ l_{2,1} & l_{2,2} & \dots & l_{2,m} \\ \vdots & \vdots & & \vdots \\ l_{m,1} & l_{m,2} & \dots & l_{m,m} \end{pmatrix} + (b_1, \dots, b_m).$$

Przypuśćmy, że Oskar dowiedział się, że tekst jawny

`adisplayedequation`

został zaszyfrowany do postaci

`DSRMSIOPXLJBZULLM`

oraz że  $m = 3$ . Oblicz klucz pokazujący wszystkie wykonane obliczenia.

- 1.6. Pokażemy, jak można dokonać kryptanalizy szyfru Hilla, używając do tego celu znanego tekstu zaszyfrowanego. Założmy, że znamy wartość  $m: m = 2$ . Podzielmy tekst zaszyfrowany na bloki dwuliterowe (digramy). Każdy taki digram kryje odpowiedni digram tekstu jawnego, zaszyfrowanego za pomocą nieznanej macierzy szyfrowania. Wybieramy najczęściej występujący digram tekstu zaszyfrowanego, zakładając, że reprezentuje on pewien często pojawiający się digram z listy w tablicy 1.1 (na przykład *TH* lub *ST*). Dla każdej takiej hipotezy postępujemy tak jak w przypadku ataku na podstawie znanego tekstu jawnego, aż znajdziemy właściwą macierz szyfrowania.

Odczytaj tą metodą następujący tekst zaszyfrowany:

`LMQETXYEAGTXCTUIEWNCXLZEWAISPZYVAPEWLMGQWYA  
XFTCJMSQCADAGTXLMDXNSNPJQSYVAPRIQSMHNOCVAXFV`

- 1.7. Opiszmy tu szczególny przypadek szyfru permutującego. Niech  $m, n$  będą dodatnimi liczbami całkowitymi. Wypisujemy tekst jawny w wierszach, tworząc prostokąty  $m \times n$ . Następnie sporządzamy tekst zaszyfrowany, wypisując znaki kolumna po kolumnie. Na przykład, gdy  $m = 4, n = 3$ , tekst jawny „cryptography” zapisalibyśmy tak:

`cryp  
togr  
aphy`

a wówczas tekst zaszyfrowany wyglądałby tak: „CTAROPYGHPRY”.

- (a) Opisz, jak powinien postępować Bolek podczas deszyfrowania tego tekstu (z danymi wartościami  $m$  i  $n$ ).
- (b) Odczytaj natępujący tekst zaszyfrowany, otrzymany w wyniku zastosowania takiej metody szyfrowania:

`MYAMRARUYIQTENCTORAHROYWDSOYEYOUARRGDERNOGW`

- 1.8. Nad  $\mathbb{Z}_2$  istnieje osiem różnych rekurencji liniowych stopnia 4, dla których  $c_0 = 1$ . Określ, które z nich prowadzą do klucza strumieniowego o okresie równym 15 (dla danego niezerowego wektora początkowego).

- 1.9. W tym ćwiczeniu udowodnimy odwracalność macierzy współczynników  $m \times m$  z punktu 1.2.5. Warunek ten jest równoważny liniowej niezależności wierszy tej macierzy nad  $\mathbb{Z}_2$ .

Jak poprzednio założymy, że rekurencja ma następującą postać:

$$z_{m+i} = \sum_{j=0}^{m-1} c_j z_{i+j} \pmod{2}$$

z wektorem początkowym  $(z_1, \dots, z_m)$ . Dla  $i \geq 1$  niech

$$v_i = (z_i, \dots, z_{i+m-1}).$$

Zauważmy, że wierszami macierzy współczynników są wektory  $v_1, \dots, v_m$ . Należy zatem udowodnić, że wektory te stanowią układ liniowo niezależny.

Udowodnij, że

- (a) Jeśli  $i \geq 1$ , to

$$v_{m+i} = \sum_{j=0}^{m-1} c_j v_{i+j} \pmod{2}.$$

- (b) Jeśli  $h$  jest najmniejszą liczbą całkowitą, taką że istnieje nietrywialna kombinacja liniowa wektorów  $v_1, \dots, v_h$  równa wektorowi zerowemu  $(0, \dots, 0)$  modulo 2, to

$$v_h = \sum_{j=0}^{h-2} \alpha_j v_{j+1} \pmod{2},$$

przy czym nie wszystkie współczynniki  $\alpha_j$  są zerami. Zauważ, że  $h \leq m+1$ , ponieważ dowolny układ  $m+1$  wektorów w  $m$ -wymiarowej przestrzeni liniowej jest liniowo zależny.

- (c) Klucz strumieniowy musi spełniać zależność rekurencyjną

$$z_{h-1+i} = \sum_{j=0}^{h-2} \alpha_j z_{j+i} \pmod{2}$$

dla każdego  $i \geq 1$ .

- (d) Jeśli  $h \leq m$ , to klucz strumieniowy spełnia zależność rekurencyjną stopnia mniejszego niż  $m$ , co jest sprzecznością. Stąd  $h = m+1$  i macierz musi być odwracalna.

- 1.10. Stosując wyczerpujące przeszukiwanie kluczy, odczytaj następujący tekst zaszyfrowany za pomocą szyfru z automatycznym generowaniem klucza:

MALVVMAFBHBUQPTSOXALTGVWWRG

- 1.11. Zajmiemy się teraz szyfrem strumieniowym będącym modyfikacją **szyfru Vi-genèra**a. Dla danego słowa kluczowego  $(K_1, \dots, K_m)$  o długości  $m$  utwórzmy klucz strumieniowy, stosując następującą regułę:  $z_i = K_i$ , gdy  $1 \leq i \leq m$ , oraz

$z_{i+m} = z_i + 1 \bmod 26$ , gdy  $i \geq m+1$ . Innymi słowy, przy każdym użyciu słowa kluczowego zastępujemy każdą literę jej następcą modulo 26. Na przykład, jeśli słowem kluczowym jest *SUMMER*, to do zaszyfrowania pierwszych sześciu liter używamy *SUMMER*, dla następnych sześciu *TVNNFS* itd.

Opisz, jak można wykorzystać pojęcie indeksu zgodności do ustalenia, po pierwsze, długości słowa kluczowego, a po drugie, samego słowa kluczowego.

Sprawdź swoją metodę, przeprowadzając kryptoanalizę następującego tekstu zaszyfrowanego:

IYMSILONRFCQXQJEDSHBUIBCJUZBOLFQYSCHATPEQGQ  
JEJNGNXZWHHGWFSSUKULJQACZKKJOAAHGKEMTAFGMKVRDO  
PXNEHEKZNKFSKIFRQVHHOVXINPHMRTJPYWQGJWPDUUVKFP  
OAWPMRKKQZWLQDYAZDRMLPBJKJOBWIWPSEPVQMBCRYVC  
RUZAAQUMBCHDAGDIEMSZFZHAGICKEMJJFPCIWKRMLMPIN  
AYOFIREAOLDTHITDVRMSE

Tekst jawnny pochodzi z „The Codebreakers” D. Kahna, Macmillan, 1967.

# 2

---

## Teoria Shannona

---

W 1949 roku Claude Shannon opublikował w czasopiśmie *Bell Systems Technical Journal* pracę zatytułowaną „Communication Theory of Secrecy Systems”. Artykuł wywarł ogromny wpływ na naukowe badania kryptograficzne. W niniejszym rozdziale omówimy niektóre koncepcje Shannona.

---

### 2.1. Tajność doskonała

Istnieją dwa zasadnicze podejścia do kwestii bezpieczeństwa kryptosystemu.

**Bezpieczeństwo obliczeniowe.** Miarą bezpieczeństwa obliczeniowego jest wysiłek obliczeniowy niezbędny do złamania systemu kryptograficznego. Możemy określić system kryptograficzny jako *obliczeniowo bezpieczny*, jeśli najlepszy algorytm, jakim można się posłużyć do jego złamania, wymaga wykonania co najmniej  $N$  operacji, gdzie  $N$  jest pewną ustaloną, bardzo dużą liczbą. Trudność takiej definicji polega jednak na tym, że o żadnym stosowanym systemie kryptograficznym nie można stwierdzić, iż spełnia jej warunek. Stąd w praktyce określa się zwykle system kryptograficzny jako „*obliczeniowo bezpieczny*”, gdy najlepsza znana metoda jego złamania wymaga nieosiągalnie długiego czasu komputerowego (co, rzecz jasna, jest warunkiem zupełnie innym niż istnienie dowodu bezpieczeństwa). Inne podejście polega na sprowadzeniu kwestii obliczeniowego bezpieczeństwa systemu kryptograficznego do pewnego dobrze zbadanego problemu, o którym wiadomo, że jest bardzo trudny. Na przykład, może to oznaczać udowodnienie zdania „dany system kryptograficzny jest bezpieczny, jeśli danej liczby całkowitej  $n$  nie można rozłożyć na czynniki pierwsze”. Takie systemy kryptograficzne nazywa się niekiedy systemami „*dowodliwie bezpiecznymi*”. Należy jednak zdawać sobie sprawę z tego, że przy takim podejściu uzyskuje się jedynie dowód bezpieczeństwa systemu względem innego problemu, nie zaś absolutny dowód bezpieczeństwa.<sup>1</sup>

<sup>1</sup>Ta sytuacja przypomina dowód NP-zupełności: wykazuje on tylko to, że dany problem jest co najmniej równie trudny jak dowolny inny problem NP-zupełny, nie dostarcza natomiast absolutnego dowodu jego złożoności obliczeniowej.

**Bezpieczeństwo bezwarunkowe.** Ta miara odnosi się do bezpieczeństwa systemu kryptograficznego, gdy nie ma ograniczeń na długość obliczeń, jakie może wykonywać Oskar. System określamy jako *bezwarkunkowo bezpieczny*, jeśli nie można go złamać, dysponując nawet nieskończonymi zasobami obliczeniowymi.

Gdy mówimy o bezpieczeństwie systemu, powinniśmy ustalić, jaki typ ataku chcemy rozważyć. W rozdziale 1 widzieliśmy, że ani szyfr z przesunięciem, ani szyfr podstawieniowy, ani szyfr Vigenèrea nie zapewniają bezpieczeństwa obliczeniowego wobec ataku opartego na znany tekście zaszyfrowanym (jeśli tekst ten jest dostatecznie długi).

W tym podrozdziale opiszemy teorię systemów kryptograficznych bezwarunkowo bezpiecznych na wypadek ataku ze znany tekstem zaszyfrowanym. Okazuje się, że wszystkie trzy wymienione wyżej szyfry są bezwarunkowo bezpieczne, jeśli do każdego elementu tekstu jawnego użyto innego klucza!

Oczywiście trudno oceniać bezwarunkowe bezpieczeństwo z punktu widzenia złożoności obliczeniowej, gdyż dopuszczały nieskończony czas obliczenia. Odpowiednim narzędziem do badania takiego typu bezpieczeństwa jest natomiast teoria prawdopodobieństwa. Potrzebne nam będą jedynie elementarne fakty z tej dziedziny. Przypomnijmy najpierw podstawowe definicje.

### DEFINICJA 2.1

Niech  $\mathbf{X}$  i  $\mathbf{Y}$  będą zmiennymi losowymi. Prawdopodobieństwo przyjęcia przez zmienną  $\mathbf{X}$  wartości  $x$  oznaczamy przez  $p(x)$ , a prawdopodobieństwo przyjęcia przez zmienną  $\mathbf{Y}$  wartości  $y$  przez  $p(y)$ . *Prawdopodobieństwo łączne*  $p(x,y)$  określa prawdopodobieństwo jednoczesnego przyjęcia przez zmienną  $\mathbf{X}$  wartości  $x$  oraz przez zmienną  $\mathbf{Y}$  wartości  $y$ . *Prawdopodobieństwo warunkowe*  $p(x|y)$  określa prawdopodobieństwo przyjęcia przez  $\mathbf{X}$  wartości  $x$ , jeśli zmienna  $\mathbf{Y}$  przyjęła wartość  $y$ . Zmienne losowe  $\mathbf{X}$  oraz  $\mathbf{Y}$  są *niezależne*, jeśli  $p(x,y) = p(x)p(y)$  dla wszystkich wartości  $x$  zmiennej  $\mathbf{X}$  i  $y$  zmiennej  $\mathbf{Y}$ .

Prawdopodobieństwo łączne jest związane z prawdopodobieństwem warunkowym następującą zależnością:

$$p(x,y) = p(x|y)p(y).$$

Zamieniając miejscami  $x$  i  $y$ , mamy równość

$$p(x,y) = p(y|x)p(x).$$

Z tych dwóch wzorów otrzymujemy natychmiast następujący wynik, znany jako twierdzenie Bayesa:

### TWIERDZENIE 2.1 (twierdzenie Bayesa)

Jeśli  $p(y) > 0$ , to

$$p(x|y) = \frac{p(x)p(y|x)}{p(y)}.$$

**WNIOSEK 2.2**

Zmienne  $\mathbf{X}$  i  $\mathbf{Y}$  są niezależne wtedy i tylko wtedy, gdy  $p(x|y) = p(x)$  dla wszystkich wartości  $x$  i  $y$ .

W całym tym podrozdziale zakładamy, że każdy klucz stosuje się tylko raz. Przypuśćmy, że mamy pewien rozkład prawdopodobieństwa na przestrzeni tekstu jawnego  $\mathcal{P}$ . Niech  $p_{\mathcal{P}}(x)$  oznacza prawdopodobieństwo *a priori* wystąpienia tekstu jawnego  $x$ . Przymajemy też, że Alicja i Bolek wybierają klucz  $K$  zgodnie z pewnym ustalonym rozkładem probabilistycznym (często klucz wybierany jest losowo, więc wszystkie klucze są równie prawdopodobne, lecz na ogół tak być nie musi). Niech  $p_{\mathcal{K}}(K)$  oznacza prawdopodobieństwo wyboru klucza  $K$ . Przyпомнijmy, że przy wyborze klucza Alicja nie zna jeszcze tekstu jawnego, stąd rozsądne wydaje się przyjęcie założenia, że wybór klucza  $K$  i tekstu jawnego  $x$  są zdarzeniami niezależnymi.

Dwa rozkłady prawdopodobieństwa na  $\mathcal{P}$  i  $\mathcal{K}$  indukują rozkład prawdopodobieństwa także na  $\mathcal{C}$ . Istotnie, nietrudno obliczyć prawdopodobieństwo  $p_{\mathcal{C}}(y)$  tego, że przekazywany jest właśnie tekst zaszyfrowany  $y$ . Dla klucza  $K \in \mathcal{K}$  niech

$$C(K) = \{e_K(x) : x \in \mathcal{P}\}.$$

Tak więc  $C(K)$  reprezentuje zbiór wszystkich możliwych tekstów zaszyfrowanych, które można otrzymać za pomocą klucza  $K$ . Wówczas dla każdego  $y \in \mathcal{C}$  mamy

$$p_{\mathcal{C}}(y) = \sum_{\{K: y \in C(K)\}} p_{\mathcal{K}}(K) p_{\mathcal{P}}(d_K(y)).$$

Zauważmy także, że dla  $y \in \mathcal{C}$  i  $x \in \mathcal{P}$  możemy obliczyć prawdopodobieństwo warunkowe  $p_{\mathcal{C}}(y|x)$ , czyli prawdopodobieństwo tego, że tekstem zaszyfrowanym będzie  $y$ , gdy tekstem jawnym jest  $x$ :

$$p_{\mathcal{C}}(y|x) = \sum_{\{K: x = d_K(y)\}} p_{\mathcal{K}}(K).$$

Teraz za pomocą twierdzenia Bayesa można obliczyć prawdopodobieństwo warunkowe  $p_{\mathcal{P}}(x|y)$ , czyli prawdopodobieństwo tego, że tekstem jawnym był  $x$ , jeśli tekstem zaszyfrowanym jest  $y$ . Otrzymujemy następujący wzór:

$$p_{\mathcal{P}}(x|y) = \frac{p_{\mathcal{P}}(x) \sum_{\{K: x = d_K(y)\}} p_{\mathcal{K}}(K)}{\sum_{\{K: y \in C(K)\}} p_{\mathcal{K}}(K) p_{\mathcal{P}}(d_K(y))}.$$

Zauważmy, że te obliczenia może przeprowadzić każdy, kto zna rozkłady prawdopodobieństwa.

Popatrzmy na prosty przykład obliczania omawianych rozkładów.

**Przykład 2.1**

Niech  $\mathcal{P} = \{a, b\}$  oraz  $p_{\mathcal{P}}(a) = 1/4$ ,  $p_{\mathcal{P}}(b) = 3/4$ . Niech dalej  $K = \{K_1, K_2, K_3\}$ ,  $p_K(K_1) = 1/2$ ,  $p_K(K_2) = p_K(K_3) = 1/4$  oraz  $\mathcal{C} = \{1, 2, 3, 4\}$ . Załóżmy, że funkcje szyfrowania są określone następująco:  $e_{K_1}(a) = 1$ ,  $e_{K_1}(b) = 2$ ;  $e_{K_2}(a) = 2$ ,  $e_{K_2}(b) = 3$  oraz  $e_{K_3}(a) = 3$ ,  $e_{K_3}(b) = 4$ . Ten system kryptograficzny można opisać następującą macierzą szyfrowania:

	$a$	$b$
$K_1$	1	2
$K_2$	2	3
$K_3$	3	4

Obliczmy rozkład prawdopodobieństwa  $p_{\mathcal{C}}$ . Otrzymujemy

$$\begin{aligned} p_{\mathcal{C}}(1) &= \frac{1}{8} \\ p_{\mathcal{C}}(2) &= \frac{3}{8} + \frac{1}{16} = \frac{7}{16} \\ p_{\mathcal{C}}(3) &= \frac{3}{16} + \frac{1}{16} = \frac{1}{4} \\ p_{\mathcal{C}}(4) &= \frac{3}{16}. \end{aligned}$$

Teraz możemy obliczyć rozkłady prawdopodobieństw warunkowych na tekście jawnym przy założeniu, że natrafiono na określony tekst zaszyfrowany. Mamy więc:

$$\begin{array}{ll} p_{\mathcal{P}}(a|1) = 1 & p_{\mathcal{P}}(b|1) = 0 \\ p_{\mathcal{P}}(a|2) = \frac{1}{7} & p_{\mathcal{P}}(b|2) = \frac{6}{7} \\ p_{\mathcal{P}}(a|3) = \frac{1}{4} & p_{\mathcal{P}}(b|3) = \frac{3}{4} \\ p_{\mathcal{P}}(a|4) = 0 & p_{\mathcal{P}}(b|4) = 1. \end{array}$$

□

Jesteśmy już przygotowani do zdefiniowania pojęcia tajności doskonałej. Nieformalnie, tajność doskonałą osiąga się wtedy, gdy Oskar nie może uzyskać żadnych informacji o tekście jawnym na podstawie obserwacji tekstu zaszyfrowanego. Do uściślenia tego pojęcia wykorzystamy znane już rozkłady prawdopodobieństwa.

**DEFINICJA 2.2**

System kryptograficzny jest *doskonale tajny*, gdy  $p_{\mathcal{P}}(x|y) = p_{\mathcal{P}}(x)$  dla każdego  $x \in \mathcal{P}$  i  $y \in \mathcal{C}$ . Jest więc tak wtedy, gdy prawdopodobieństwo *a posteriori* tego, że tekstem jawnym jest  $x$ , jeśli tekstem zaszyfrowanym jest  $y$ , jest równe prawdopodobieństwu *a priori* tego, że tekstem jawnym jest  $x$ .

W przykładzie 2.1 mamy tajność doskonałą dla tekstu zaszyfrowanego 3, ale nie dla pozostałych trzech tekstów zaszyfrowanych.

Udowodnimy teraz, że szyfr z przesunięciem zapewnia tajność doskonałą. Intuicyjnie wydaje się to oczywiste, gdyż jeśli  $y \in \mathbb{Z}_{26}$  jest tekstem zaszyfrowanym, to przecież może on reprezentować dowolny element  $x \in \mathbb{Z}_{26}$ , w zależności od klucza. Następne twierdzenie, w którego dowodzie wykorzystamy rozkłady prawdopodobieństw, daje formalny wyraz tej sytuacji.

### TWIERDZENIE 2.3

Załóżmy, że każdy z 26 kluczów szyfru z przesunięciem jest używany z jednakowym prawdopodobieństwem  $1/26$ . Wówczas dla dowolnego rozkładu na przestrzeni tekstu jawnego szyfr z przesunięciem zapewnia tajność doskonałą.

**DOWÓD** Przypomnijmy, że  $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}$  oraz że dla  $0 \leq K \leq 25$  reguła szyfrowania  $e_K$  jest określona następująco:  $e_K(x) = x + K \pmod{26}$  ( $x \in \mathbb{Z}_{26}$ ). Obliczmy najpierw rozkład  $p_C$ . Niech  $y \in \mathbb{Z}_{26}$ . Wówczas

$$\begin{aligned} p_C(y) &= \sum_{K \in \mathbb{Z}_{26}} p_K(K) p_{\mathcal{P}}(d_K(y)) \\ &= \sum_{K \in \mathbb{Z}_{26}} \frac{1}{26} p_{\mathcal{P}}(y - K) \\ &= \frac{1}{26} \sum_{K \in \mathbb{Z}_{26}} p_{\mathcal{P}}(y - K). \end{aligned}$$

Dla ustalonego  $y$  wartości  $y - K \pmod{26}$  stanowią permutację zbioru  $\mathbb{Z}_{26}$ , a  $p_{\mathcal{P}}$  jest rozkładem prawdopodobieństwa. Mamy więc

$$\begin{aligned} \sum_{K \in \mathbb{Z}_{26}} p_{\mathcal{P}}(y - K) &= \sum_{K \in \mathbb{Z}_{26}} p_{\mathcal{P}}(y) \\ &= 1. \end{aligned}$$

W rezultacie

$$p_C(y) = \frac{1}{26}$$

dla każdego  $y \in \mathbb{Z}_{26}$ .

Dalej mamy

$$\begin{aligned} p_C(y|x) &= p_K(y - x \pmod{26}) \\ &= \frac{1}{26} \end{aligned}$$

dla dowolnych  $x, y$ , gdyż dla dowolnych  $x, y$  jedynym kluczem  $K$ , takim że  $e_K(x) = y$ , jest  $K = y - x \pmod{26}$ . Korzystając z twierdzenia Bayesa, dochodzimy już łatwo do równości

$$\begin{aligned}
 p_{\mathcal{P}}(x|y) &= \frac{p_{\mathcal{P}}(x)p_{\mathcal{C}}(y|x)}{p_{\mathcal{C}}(y)} \\
 &= \frac{p_{\mathcal{P}}(x)\frac{1}{26}}{\frac{1}{26}} \\
 &= p_{\mathcal{P}}(x),
 \end{aligned}$$

co dowodzi tajności doskonałej. ■

Tak więc szyfr z przesunięciem jest „nie do złamania”, jeśli do każdego znaku tekstu jawnego użyjemy innego, losowo wybranego klucza.

Przyjrzyjmy się tajności doskonałej w ogólniejszym kontekście. Po pierwsze, zauważmy, że dzięki twierdzeniu Bayesa, warunek  $p_{\mathcal{P}}(x|y) = p_{\mathcal{P}}(x)$  dla wszystkich  $x \in \mathcal{P}$ ,  $y \in \mathcal{C}$  jest równoważny warunkowi  $p_{\mathcal{C}}(y|x) = p_{\mathcal{C}}(y)$  dla wszystkich  $x \in \mathcal{P}$ ,  $y \in \mathcal{C}$ . Przyjmijmy teraz rozsądne założenie, że  $p_{\mathcal{C}}(y) > 0$  dla wszystkich  $y \in \mathcal{C}$  ( $p_{\mathcal{C}}(y) = 0$  oznacza, że tekst zaszyfrowany  $y$  wcale nie jest używany i można go w związku z tym usunąć ze zbioru  $\mathcal{C}$ ). Ustalmy element  $x \in \mathcal{P}$ . Dla każdego  $y \in \mathcal{C}$  mamy  $p_{\mathcal{C}}(y|x) = p_{\mathcal{C}}(y) > 0$ , zatem dla każdego  $y \in \mathcal{C}$  musi istnieć co najmniej jeden taki klucz  $K$ , że  $e_K(x) = y$ . Wynika stąd, że  $|\mathcal{K}| \geq |\mathcal{C}|$ . W każdym systemie kryptograficznym musi być spełniona nierówność  $|\mathcal{C}| \geq |\mathcal{P}|$ , gdyż każda reguła szyfrowania jest różnowartościowa. W przypadku skrajnym, gdy  $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{P}|$ , potrafimy podać ładną charakteryzację sytuacji zapewniających tajność doskonałą. Jej autorem jest C. Shannon.

#### **TWIERDZENIE 2.4**

Niech  $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$  będzie systemem kryptograficznym, takim że  $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{P}|$ . System ten zapewnia tajność doskonałą wtedy i tylko wtedy, gdy każdy klucz jest stosowany z jednakowym prawdopodobieństwem  $1/|\mathcal{K}|$  oraz dla każdego  $x \in \mathcal{P}$  i każdego  $y \in \mathcal{C}$  istnieje tylko jeden klucz  $K$ , taki że  $e_K(x) = y$ .

**DOWÓD** Założymy, że dany system kryptograficzny zapewnia tajność doskonałą. Jak stwierdziliśmy wyżej, dla każdego  $x \in \mathcal{P}$  i  $y \in \mathcal{C}$  musi istnieć co najmniej jeden taki klucz  $K$ , że  $e_K(x) = y$ . Mamy zatem nierówność:

$$\begin{aligned}
 |\mathcal{C}| &= |\{e_K(x): K \in \mathcal{K}\}| \\
 &\leq |\mathcal{K}|.
 \end{aligned}$$

Założyliśmy jednak, że  $|\mathcal{C}| = |\mathcal{K}|$ , musi zatem mieć miejsce równość

$$|\{e_K(x): K \in \mathcal{K}\}| = |\mathcal{K}|.$$

To oznacza, że nie istnieją dwa różne klucze  $K_1$  i  $K_2$ , takie że  $e_{K_1}(x) = e_{K_2}(x) = y$ . Wykazaliśmy w ten sposób, że dla każdego  $x \in \mathcal{P}$  i  $y \in \mathcal{C}$  istnieje dokładnie jeden taki klucz  $K$ , że  $e_K(x) = y$ .

Niech  $n = |\mathcal{K}|$ . Założymy, że  $\mathcal{P} = \{x_i : 1 \leq i \leq n\}$ . Dla ustalonego  $y \in \mathcal{C}$  możemy tak ponumerować klucze  $K_1, K_2, \dots$ , żeby otrzymać równości  $e_{K_i}(x_i) = y$ ,  $1 \leq i \leq n$ . Korzystając znów z twierdzenia Bayesa, mamy

$$\begin{aligned} p_{\mathcal{P}}(x_i|y) &= \frac{p_{\mathcal{C}}(y|x_i)p_{\mathcal{P}}(x_i)}{p_{\mathcal{C}}(y)} \\ &= \frac{p_{\mathcal{K}}(K_i)p_{\mathcal{P}}(x_i)}{p_{\mathcal{C}}(y)}. \end{aligned}$$

Z warunku tajności doskonałej, czyli z równości  $p_{\mathcal{P}}(x_i|y) = p_{\mathcal{P}}(x_i)$ , wynika, że  $p_{\mathcal{K}}(K_i) = p_{\mathcal{C}}(y)$  dla  $1 \leq i \leq n$ . Oznacza to, że klucze są stosowane z jednakowym prawdopodobieństwem (mianowicie równym  $p_{\mathcal{C}}(y)$ ). Ponieważ jednak liczba kluczy jest równa  $|\mathcal{K}|$ , musi być spełniona równość  $p_{\mathcal{K}}(K) = 1/|\mathcal{K}|$  dla każdego klucza  $K \in \mathcal{K}$ .

W drugą stronę, założymy, że spełnione są oba warunki, o których mówi twierdzenie. Wtedy system kryptograficzny, jak łatwo zauważyc, zapewnia tajność doskonałą dla dowolnego rozkładu prawdopodobieństwa na przestrzeni tekstów jawnych, podobnie jak w dowodzie twierdzenia 2.3. Szczegóły pozostawiamy Czytelnikowi. ■

Jedną z dobrze znanych realizacji postulatu tajności doskonałej jest **szyfr Vernama z kluczem jednorazowym**, opisany po raz pierwszy przez Gilberta Vernama w 1917 roku i stosowany do automatycznego szyfrowania i deszyfrowania wiadomości telegraficznych. Ciekawe, że przez wiele lat uznawano, że szyfr z kluczem jednorazowym jest systemem kryptograficznym „nie do złamania”, choć nie potrafiono podać na to żadnego dowodu, dopóki Shannon nie rozwiniął pojęcia tajności doskonałej ponad 30 lat później.

Opis szyfru z kluczem jednorazowym znajduje się na rysunku 2.1.

Posługując się twierdzeniem 2.4, łatwo zauważyc, że szyfr ten zapewnia tajność doskonałą. Jest to system atrakcyjny także dzięki łatwości, z jaką dokonuje się w nim szyfrowania i deszyfrowania.

Niech  $n \geq 1$  będzie liczbą całkowitą i niech  $\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbb{Z}_2)^n$ . Dla  $K \in (\mathbb{Z}_2)^n$  definiujemy  $e_K(x)$  jako sumę wektorową modulo 2 z  $K$  i  $x$  (lub równoważnie jako różnicę symetryczną dwóch odpowiednich ciągów bitów). Tak więc, jeśli  $x = (x_1, \dots, x_n)$  i  $K = (K_1, \dots, K_n)$ , to

$$e_K(x) = (x_1 + K_1, \dots, x_n + K_n) \text{ mod } 2.$$

Funkcja deszyfrowania jest identyczna jak funkcja szyfrowania. Jeśli  $y = (y_1, \dots, y_n)$ , to

$$d_K(y) = (y_1 + K_1, \dots, y_n + K_n) \text{ mod } 2.$$

**RYSUNEK 2.1.** Szyfr z kluczem jednorazowym

Vernam opatentował swój system w nadziei, że znajdzie on szerokie zastosowanie handlowe. Niestety, systemy kryptograficzne bezwarunkowo bezpieczne, takie jak szyfr z kluczem jednorazowym, mają jedną poważną wadę. Spełnienie warunku  $|K| \geq |P|$  oznacza, że ilość informacji o kluczach, jaką należy bezpiecznie przekazać, jest co najmniej równie duża, jak wielkość samego tekstu jawnego. Na przykład w przypadku szyfru z kluczem jednorazowym do zaszyfrowania  $n$  bitów tekstu jawnego potrzeba  $n$  bitów klucza. Nie stanowiłoby to dużego problemu, gdyby ten sam klucz mógł być używany do szyfrowania różnych komunikatów – ale bezpieczeństwo bezwarunkowo bezpiecznych systemów kryptograficznych wynika właśnie z tego, że każdy klucz służy do zaszyfrowania tylko jednego znaku (stąd określenie „jednorazowy” w nazwie).

Na przykład, szyfr z kluczem jednorazowym jest podatny na ataki oparte na znanym tekście jawnym, ponieważ klucz  $K$  można wyliczyć jako różnicę symetryczną ciągów bitów  $x$  i  $e_K(x)$ . Dlatego dla każdego wysyłanego komunikatu trzeba wygenerować i przesyłać bezpiecznym kanałem nowy klucz. Stwarza to poważne problemy w zakresie zarządzania kluczami, co ograniczyło stosowalność szyfru z kluczem jednorazowym w zastosowaniach handlowych; znajduje on natomiast zastosowanie w sferze militarnej i dyplomatycznej, gdzie bezwarunkowe bezpieczeństwo może mieć ogromne znaczenie.

Historyczne rozwój kryptografii zmierzał w kierunku stworzenia systemów kryptograficznych, w których ten sam klucz mógłby być używany do szyfrowania stosunkowo długiego ciągu tekstu jawnego (a więc jeden klucz do wielu komunikatów), przy zachowaniu (co najmniej) bezpieczeństwa obliczeniowego. Jednym z takich systemów jest DES (skrót od *Data Encryption Standard* – standard szyfrowania danych), którym zajmiemy się w rozdziale 3.

## 2.2. Entropia

W poprzednim podrozdziale omawialiśmy pojęcie bezpieczeństwa doskonałego. Ograniczyliśmy się do sytuacji, w której klucz jest używany tylko do jednego szyfrowania. Teraz przyjrzymy się temu, co się dzieje, gdy coraz więcej tekstów jawnych szyfruje się tym samym kluczem i spróbujemy ocenić prawdopodobieństwo sukcesu ataku opartego na znajomości tekstu zaszyfrowanego, zakładając, że kryptoanalyst dysponuje wystarczająco długim czasem.

Podstawowym narzędziem w tym badaniu jest pojęcie entropii, pochodzące z teorii informacji wprowadzonej przez Shannona w 1948 roku. Entropię można sobie wyobrażać jako matematyczną miarę informacji lub niejednoznaczności, a oblicza się ją jako pewną funkcję rozkładu prawdopodobieństwa.

Przypuśćmy, że mamy zmienną losową  $\mathbf{X}$  przyjmującą skończenie wiele wartości zgodnie z rozkładem prawdopodobieństwa  $p(\mathbf{X})$ . Jakiej informacji dostarcza zdarzenie zachodzące zgodnie z rozkładem  $p(\mathbf{X})$ ? Równoważnie, jeśli zdarzenie (jeszcze) nie zaszło, jak ocenić niepewność co do jego wyniku? Taką wielkość nazywa się entropią zmiennej losowej  $\mathbf{X}$  i oznacza symbolem  $H(\mathbf{X})$ .

Jeśli wydaje się to nazbyt abstrakcyjne, popatrzmy na konkretny przykład. Niech zmienna losowa  $\mathbf{X}$  opisuje rzut monetą. Rozkład prawdopodobieństwa wygląda tak:  $p(\text{orzeł}) = p(\text{reszka}) = 1/2$ . Rozsądny wydaje się przyjęcie, że informacja lub entropia, jaką niesie jeden rzut monetą, jest równa jednemu bitowi, gdyż moglibyśmy, na przykład, użyć 1 do zakodowania orła oraz 0 do zakodowania reszki. Podobnie entropię  $n$  niezależnych rzutów określilibyśmy jako  $n$ , gdyż  $n$  rzutów monetą można zakodować ciągiem bitów o długości  $n$ .

A oto nieco bardziej skomplikowany przykład. Niech  $\mathbf{X}$  będzie zmienną losową, przyjmującą tylko trzy wartości  $x_1, x_2, x_3$  z prawdopodobieństwem  $1/2, 1/4, 1/4$ , odpowiednio. Najbardziej efektywnym „kodowaniem” możliwych wyników jest przypisanie wynikowi  $x_1$  liczby 0, wynikowi  $x_2$  liczby 10 oraz wynikowi  $x_3$  liczby 11. Wówczas średnią liczbę bitów potrzebną do zakodowania zmiennej  $\mathbf{X}$  jest

$$\frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 2 = \frac{3}{2}.$$

Powyższe przykłady sugerują, że zdarzenie o prawdopodobieństwie  $2^{-n}$  można zakodować ciągiem bitów o długości  $n$ . Ogólniej, możemy sobie wyobrażać, że zdarzenie zachodzące z prawdopodobieństwem  $p$  można zakodować za pomocą ciągu bitów o długości równej w przybliżeniu  $-\log_2 p$ . Mając dany dowolny rozkład prawdopodobieństwa  $p_1, p_2, \dots, p_n$  zmiennej losowej  $\mathbf{X}$ , przyjmiemy jako miarę informacji średnią ważoną wielkości  $-\log_2 p_i$ . Taka motywacja prowadzi do następującej definicji formalnej.

### DEFINICJA 2.3

Niech  $\mathbf{X}$  będzie zmienną losową przyjmującą tylko skończenie wiele wartości zgodnie z rozkładem prawdopodobieństwa  $p(\mathbf{X})$ . Wówczas entropią tego rozkładu nazwiemy liczbę

$$H(\mathbf{X}) = - \sum_{i=1}^n p_i \log_2 p_i.$$

Jeśli  $x_i$  dla  $1 \leq i \leq n$  są wszystkimi możliwymi wartościami zmiennej  $\mathbf{X}$ , to

$$H(\mathbf{X}) = - \sum_{i=1}^n p(\mathbf{X} = x_i) \log_2 p(\mathbf{X} = x_i).$$

**UWAGA** Gdy  $p_i = 0$ , wartość  $\log_2 p_i$  nie jest określona. Stąd definiuje się niekiedy entropię jako sumę po wszystkich niezerowych prawdopodobieństwach. Ponieważ  $\lim_{x \rightarrow 0} x \log_2 x = 0$ , dopuszczenie przypadku, gdy  $p_i = 0$  dla pewnego  $i$ , nie sprawiałoby trudności. Przyjmiemy jednak milcząco, że przy obliczaniu entropii rozkładu  $p_i$  sumujemy tylko po tych indeksach  $i$ , dla których  $p_i \neq 0$ . Dodajmy jeszcze, że wybór liczby 2 jako podstawy logarytmu jest arbitralny, gdyż przy innej podstawie wartość entropii zmieniłaby się tylko o stały czynnik. ■

Zauważmy, że gdy  $p_i = 1/n$  dla  $1 \leq i \leq n$ , wówczas  $H(\mathbf{X}) = \log_2 n$ . Podobnie łatwo stwierdzić, że  $H(\mathbf{X}) \geq 0$ , a  $H(\mathbf{X}) = 0$  wtedy i tylko wtedy, gdy  $p_i = 1$  dla pewnego  $i$  oraz  $p_j = 0$  dla wszystkich  $j \neq i$ .

Przyjrzyjmy się entropii różnych składników systemu kryptograficznego. Możemy myśleć o kluczu jako o zmiennej losowej  $\mathbf{K}$ , przyjmującej wartości zgodnie z rozkładem prawdopodobieństwa  $p_K$ , i w tym kontekście możemy obliczyć entropię  $H(\mathbf{K})$ . Podobnie możemy liczyć entropie  $H(\mathbf{P})$  i  $H(\mathbf{C})$  związane z rozkładami prawdopodobieństwa przestrzeni tekstu jawnego i tekstu zaszyfrowanego, odpowiednio.

Dla ilustracji obliczmy entropie systemu kryptograficznego z przykładu 1.2.

### Przykład 2.1 cd.

Zgodnie z definicją, obliczamy:

$$\begin{aligned} H(\mathbf{P}) &= -\frac{1}{4} \log_2 \frac{1}{4} - \frac{3}{4} \log_2 \frac{3}{4} \\ &= -\frac{1}{4}(-2) - \frac{3}{4}(\log_2 3 - 2) \\ &= 2 - \frac{3}{4}(\log_2 3) \\ &\approx 0,81. \end{aligned}$$

Z podobnych obliczeń wynika, że  $H(\mathbf{K}) = 1,5$  oraz  $H(\mathbf{P}) \approx 1,85$ . □

#### 2.2.1. Kodowania Huffmmana a entropia

Przedstawimy tu pokrótkie związki między entropią i kodowaniami Huffmanna. Wyniki z tego podrozdziału nie są istotne dla kryptograficznych zastosowań entropii, stąd można go opuścić przy pierwszym czytaniu, nie tracąc ciągłości wypowiedzi. Jednak omówione tu kwestie mogą posłużyć jako dodatkowe uzasadnienie pojęcia entropii.

Entropię wprowadziliśmy w kontekście kodowania zdarzeń losowych zachodzących zgodnie z pewnym określonym rozkładem prawdopodobieństwa. Zaczniemy zatem od uściślenia tych pojęć. Jak poprzednio, niech  $\mathbf{X}$  będzie zmienną losową przyjmującą skończenie wielu wartości i niech  $p(\mathbf{X})$  będzie związanym z nią rozkładem prawdopodobieństwa.

*Kodowaniem* zmiennej  $\mathbf{X}$  nazwiemy dowolne przekształcenie

$$f : \mathbf{X} \rightarrow \{0,1\}^*,$$

gdzie  $\{0,1\}^*$  oznacza zbiór wszystkich skończonych ciągów zero-jedynkowych. Mając skończoną listę (lub ciąg) zdarzeń  $x_1 \dots x_n$ , możemy w oczywisty sposób rozszerzyć kodowanie  $f$ , definiując

$$f(x_1 \dots x_n) = f(x_1) \| \dots \| f(x_n)$$

gdzie  $\parallel$  oznacza konkatenację ciągów. Możemy zatem traktować  $f$  jako funkcję  $f : \mathbf{X}^* \rightarrow \{0, 1\}^*$ .

Przypuśćmy, że ciąg  $x_1 \dots x_n$  jest wytwarzany przez źródło bez pamięci, tak że każde ze zdarzeń  $x_i$  występuje zgodnie z pewnym rozkładem prawdopodobieństwa zmiennej  $\mathbf{X}$ . Oznacza to, że prawdopodobieństwo wystąpienia dowolnego ciągu  $x_1 \dots x_n$  jest równe  $p(x_1) \cdot \dots \cdot p(x_n)$ . (Zauważmy, że wyrazy ciągu nie muszą być parami różne, gdyż źródło jest pozbawione pamięci; prostym przykładem jest tu ciąg  $n$  rzutów monetą<sup>†</sup>).

Jeśli teraz mamy kodować ciągi za pomocą funkcji  $f$ , warto się upewnić, że potrafimy je jednoznacznie odkodować. Stąd kodowanie  $f$  powinno być przekształceniem różnowartościowym.

### Przykład 2.2

Niech  $\mathbf{X} = \{a, b, c, d\}$ . Rozpatrzmy następujące trzy kodowania:

$$\begin{array}{llll} f(a) = 1 & f(b) = 10 & f(c) = 100 & f(d) = 1000 \\ g(a) = 0 & g(b) = 10 & g(c) = 110 & g(d) = 111 \\ h(a) = 0 & h(b) = 01 & h(c) = 10 & h(d) = 11 \end{array}$$

Widać tu, że funkcje  $f$  i  $g$  są kodowaniami różnowartościowymi, w odróżnieniu od funkcji  $h$ . Ciąg kodowany za pomocą funkcji  $f$  można odkodowywać, czytając od prawej do lewej; napotkanie 1 oznacza koniec kodu pojedynczego elementu.

Z kolei odkodowywanie ciągu, do którego użyto funkcji  $g$ , należy zaczynać od początku (od lewej) i postępować sekwencyjnie. Natrafiając na podciąg reprezentujący  $a, b, c$  lub  $d$ , odkodowujemy go i odcinamy od reszty. Na przykład, odczytując ciąg 10101110, tłumaczymy 10 jako  $b$ , następnie 10 znów jako  $b$ , następnie 111 zastępujemy przez  $d$ , wreszcie 0 zamieniamy na  $a$ . Odkodowanym ciągiem jest więc  $bbda$ .

Aby się przekonać, że  $h$  nie jest różnowartościowa, wystarczy przykład:

$$h(ac) = h(ba) = 010.$$

□

Z punktu widzenia łatwości odkodowywania funkcja  $g$  ma przewagę nad  $f$ . Pozwala ona bowiem na kolejne odkodowywanie od początku do końca, bez odwoływania się do pamięci. Proste sekwencyjne odkodowywanie funkcji  $g$  jest możliwe dzięki temu, że  $g$  jest kodowaniem wolnym od przedrostków. (Kodowanie  $g$  jest wolne od przedrostków, gdy nie istnieją dwa elementy  $x, y \in \mathbf{X}$  oraz ciąg  $z \in \{0, 1\}^*$ , takie że  $g(x) = g(y) \parallel z$ ).

W tym, o czym mówiliśmy dotąd, nie pojawiła się entropia. Czytelnik nie będzie zapewne zdziwiony, gdy stwierdzimy, że entropia wiąże się ze skutecznością

---

<sup>†</sup>W polskiej terminologii takie ciągi nazywa się *wariacjami z powtarzeniami* (przyp. tłum.).

kodowania. Skuteczność kodowania  $f$  będziemy mierzyć tak jak poprzednio: jest nią ważona średnia długości kodowania elementu  $\mathbf{X}$  (oznaczana symbolem  $\ell(f)$ ). Mamy zatem następującą definicję:

$$\ell(f) = \sum_{x \in \mathbf{X}} p(x)|f(x)|,$$

gdzie  $|y|$  oznacza długość ciągu  $y$ .

Otoż nasze podstawowe zadanie polega na znalezieniu takiego różnowartościowego kodowania  $f$ , które minimalizuje wartość  $\ell(f)$ . Istnieje dobrze znany algorytm, zwany algorytmem Huffmana, które to zadanie realizuje. Co więcej, kodowanie  $f$ , otrzymane w wyniku zastosowania algorytmu Huffmana, jest wolne od przedrostków oraz

$$H(\mathbf{X}) \leq \ell(f) < H(\mathbf{X}) + 1.$$

Tak więc wartość entropii stanowi bliskie przybliżenie średniej długości optymalnego kodowania różnowartościowego.

Nie będziemy dowodzić przytaczanych tu wyników, podamy jednak krótki, nieformalny opis algorytmu Huffmana. Na początku mamy dany rozkład prawdopodobieństwa na zbiorze  $\mathbf{X}$ , a kod każdego elementu jest pusty. W każdej iteracji dwa elementy o najmniejszym prawdopodobieństwie są łączone w jeden element, którego prawdopodobieństwo jest sumą ich prawdopodobieństw. Mniejszemu z dwóch elementów przypisujemy „0”, a większemu wartość „1”. Gdy zostanie już tylko jeden element, możemy odtworzyć kodowanie każdego  $x \in \mathbf{X}$ , posuwając się po ciągu elementów „wstecz” od elementu końcowego do początkowego elementu  $x$ .

Wyjaśni to prosty przykład.

### Przykład 2.3

Niech  $\mathbf{X} = \{a, b, c, d, e\}$  ma następujący rozkład prawdopodobieństwa:  $p(a) = 0,05$ ,  $p(b) = 0,10$ ,  $p(c) = 0,12$ ,  $p(d) = 0,13$  oraz  $p(e) = 0,60$ . Algorytm Huffmana daje następującą sekwencję:

$a$	$b$	$c$	$d$	$e$
0,05	0,10	0,12	0,13	0,60
0	1			
0,15		0,12	0,13	0,60
	0	1		
0,15		0,25		0,60
0		1		
	0,40		0,60	
	0		1	
		1,0		

Prowadzi to do następującego kodowania:

$x$	$f(x)$
$a$	000
$b$	001
$c$	010
$d$	011
$e$	1

Stąd otrzymujemy średnią długość kodu:

$$\begin{aligned}\ell(f) &= 0,5 \cdot 3 + 0,10 \cdot 3 + 0,12 \cdot 3 + 0,13 \cdot 3 + 0,60 \cdot 1 \\ &= 1,8.\end{aligned}$$

Porównajmy to z entropią:

$$\begin{aligned}H(\mathbf{X}) &= 0,2161 + 0,3322 + 0,3671 + 0,3842 + 0,4422 \\ &= 1,7402.\end{aligned}$$

□

### 2.3. Własności entropii

Udowodnimy tu pewne zasadnicze wyniki dotyczące entropii. Najpierw sformułujemy bardzo użyteczną podstawową własność, znaną jako nierówność Jensena. Nierówność Jensena dotyczy funkcji wklęszych, które teraz zdefiniujemy.

#### DEFINICJA 2.4<sup>†</sup>

Funkcja rzeczywista  $f$  jest *wklęsła* w przedziale  $I$ , gdy

$$f\left(\frac{x+y}{2}\right) \geq \frac{f(x) + f(y)}{2}$$

dla wszystkich  $x, y \in I$ . Funkcja  $f$  jest *ściśle wklęsła* w przedziale  $I$ , gdy

$$f\left(\frac{x+y}{2}\right) > \frac{f(x) + f(y)}{2}$$

dla wszystkich  $x, y \in I$ , takich że  $x \neq y$ .

A oto nierówność Jensena, którą przedstawimy bez dowodu.

<sup>†</sup>Odwracając w tej definicji obie nierówności, otrzymujemy definicję funkcji wypukłej i ściśle wypukłej w przedziale  $I$  (przyp. tłum.).

**TWIERDZENIE 2.5** (nierówność Jensena)

Niech  $f$  będzie ciągłą, ściśle wklęsłą funkcją określoną na przedziale  $I$ ,

$$\sum_{i=1}^n a_i = 1,$$

oraz  $a_i > 0$ ,  $1 \leq i \leq n$ . Wówczas

$$\sum_{i=1}^n a_i f(x_i) \leq f\left(\sum_{i=1}^n a_i x_i\right),$$

gdzie  $x_i \in I$ ,  $1 \leq i \leq n$ . Ponadto, równość zachodzi wtedy i tylko wtedy, gdy  $x_1 = \dots = x_n$ .

Wyrowadzimy teraz kilka wyników dotyczących entropii. W następnym twierdzeniu wykorzystamy fakt, że funkcja  $\log_2 x$  jest ściśle wklęsła na przedziale  $(0, \infty)$ . (W istocie, wynika to natychmiast z elementarnego rachunku różniczkowego, gdyż druga pochodna funkcji logarytmicznej jest ujemna w całym przedziale  $(0, \infty)$ ).

**TWIERDZENIE 2.6**

Niech  $\mathbf{X}$  będzie zmienną losową o rozkładzie prawopodobieństwa  $p_1, p_2, \dots, p_n$ , gdzie  $p_i > 0$  dla  $1 \leq i \leq n$ . Wówczas  $H(\mathbf{X}) \leq \log_2 n$ , przy czym równość ma miejsce wtedy i tylko wtedy, gdy  $p_i = 1/n$  dla wszystkich  $1 \leq i \leq n$ .

**DOWÓD** Stosując nierówność Jensena, otrzymujemy:

$$\begin{aligned} H(\mathbf{X}) &= - \sum_{i=1}^n p_i \log_2 p_i \\ &= \sum_{i=1}^n p_i \log_2 \frac{1}{p_i} \\ &\leq \log_2 \sum_{i=1}^n \left( p_i \cdot \frac{1}{p_i} \right) \\ &= \log_2 n. \end{aligned}$$

Ponadto równość ma miejsce wtedy i tylko wtedy, gdy  $p_i = 1/n$  dla każdego  $1 \leq i \leq n$ . ■

**TWIERDZENIE 2.7**

$H(\mathbf{X}, \mathbf{Y}) \leq H(\mathbf{X}) + H(\mathbf{Y})$ , przy czym równość ma miejsce wtedy i tylko wtedy, gdy  $\mathbf{X}$  i  $\mathbf{Y}$  są zmiennymi niezależnymi.

**DOWÓD** Założymy, że  $\mathbf{X}$  przyjmuje wartości  $x_i$ ,  $1 \leq i \leq m$ , a wartościami zmiennej  $\mathbf{Y}$  są  $y_j$ ,  $1 \leq j \leq n$ . Niech  $p_i = p(\mathbf{X} = x_i)$  dla  $1 \leq i \leq m$  oraz  $q_j = p(\mathbf{Y} = y_j)$  dla  $1 \leq j \leq n$ . Niech ponadto  $r_{ij} = p(\mathbf{X} = x_i, \mathbf{Y} = y_j)$ ,  $1 \leq i \leq m$ ,  $1 \leq j \leq n$  (to rozkład prawdopodobieństwa łącznego).

Zauważmy, że

$$p_i = \sum_{j=1}^n r_{ij}$$

( $1 \leq i \leq m$ ) oraz

$$q_j = \sum_{i=1}^m r_{ij}$$

( $1 \leq j \leq n$ ). Stąd obliczamy co następuje

$$\begin{aligned} H(\mathbf{X}) + H(\mathbf{Y}) &= - \left( \sum_{i=1}^m p_i \log_2 p_i + \sum_{j=1}^n q_j \log_2 q_j \right) \\ &= - \left( \sum_{i=1}^m \sum_{j=1}^n r_{ij} \log_2 p_i + \sum_{j=1}^n \sum_{i=1}^m r_{ij} \log_2 q_j \right) \\ &= - \sum_{i=1}^m \sum_{j=1}^n r_{ij} \log_2 p_i q_j. \end{aligned}$$

Jednakże

$$H(\mathbf{X}, \mathbf{Y}) = - \sum_{i=1}^m \sum_{j=1}^n r_{ij} \log_2 r_{ij}.$$

Łącząc otrzymane równości, mamy:

$$\begin{aligned} H(\mathbf{X}, \mathbf{Y}) - H(\mathbf{X}) - H(\mathbf{Y}) &= \sum_{i=1}^m \sum_{j=1}^n r_{ij} \log_2 \frac{1}{r_{ij}} + \sum_{i=1}^m \sum_{j=1}^n r_{ij} \log_2 p_i q_j \\ &= \sum_{i=1}^m \sum_{j=1}^n r_{ij} \log_2 \frac{p_i q_j}{r_{ij}} \\ &\leq \log_2 \sum_{i=1}^m \sum_{j=1}^n p_i q_j \\ &= \log_2 1 \\ &= 0. \end{aligned}$$

(Zastosowaliśmy tu nierówność Jensena i skorzystaliśmy z tego, że  $r_{ij}$  stanowią rozkład prawdopodobieństwa).

Możemy także stwierdzić, kiedy ma miejsce równość: musi istnieć stała  $c$ , taka że  $p_i q_j / r_{ij} = c$  dla wszystkich  $i, j$ . Z równości

$$\sum_{j=1}^n \sum_{i=1}^m r_{ij} = \sum_{j=1}^n \sum_{i=1}^m p_i q_j = 1$$

wnioskujemy, że  $c = 1$ . Tak więc równość ma miejsce wtedy i tylko wtedy, gdy  $r_{ij} = p_i q_j$ , czyli wtedy i tylko wtedy, gdy

$$p(\mathbf{X} = x_i, \mathbf{Y} = y_j) = p(\mathbf{X} = x_i)p(\mathbf{Y} = y_j),$$

$1 \leq i \leq m$ ,  $1 \leq j \leq n$ . Ale to oznacza właśnie, że zmienne  $\mathbf{X}$  i  $\mathbf{Y}$  są niezależne. ■

Zdefiniujemy teraz pojęcie entropii warunkowej.

### DEFINICJA 2.5

Niech  $\mathbf{X}$  i  $\mathbf{Y}$  będą zmiennymi losowymi. Wówczas dla dowolnej ustalonej wartości  $\mathbf{Y}$  mamy rozkład prawdopodobieństwa (warunkowego)  $p(\mathbf{X}|y)$ . Oczywiście,

$$H(\mathbf{X}|y) = - \sum_x p(x|y) \log_2 p(x|y).$$

Entropię warunkową  $H(\mathbf{X}|\mathbf{Y})$  określmy jako średnią ważoną (względem prawdopodobieństw  $p(y)$ ) entropii  $H(\mathbf{X}|y)$  dla wszystkich wartości  $y$ . Tak więc

$$H(\mathbf{X}|\mathbf{Y}) = - \sum_y \sum_x p(y) p(x|y) \log_2 p(x|y).$$

Entropia warunkowa jest miarą średniej ilości informacji, jakiej dostarcza  $\mathbf{Y}$  o zmiennej  $\mathbf{X}$ .

Następne dwa wyniki są oczywiste, ich dowody pozostawiamy jako ćwiczenie.

### TWIERDZENIE 2.8

$$H(\mathbf{X}, \mathbf{Y}) = H(\mathbf{X}) + H(\mathbf{X}|\mathbf{Y}).$$

### WNIOSEK 2.9

$H(\mathbf{X}|\mathbf{Y}) \leq H(\mathbf{X})$ , przy czym równość ma miejsce wtedy i tylko wtedy, gdy  $\mathbf{X}$  i  $\mathbf{Y}$  są niezależne.

## 2.4. Błędne klucze i długość krytyczna

Udowodnione wcześniej własności entropii zastosujemy tu do systemów kryptograficznych. Zaczniemy od wskazania zasadniczych relacji między entropiami składników systemu. Entropię warunkową  $H(\mathbf{K}|\mathbf{C})$  nazwiemy *niednoznacz-*

*nością klucza* (ang. *key equivocation*); stanowi ona miarę informacji, jakiej o klu-  
cze dostarcza tekst zaszyfrowany.

### TWIERDZENIE 2.10

Niech  $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$  będzie systemem kryptograficznym. Wówczas

$$H(\mathbf{K}|\mathbf{C}) = H(\mathbf{K}) + H(\mathbf{P}) - H(\mathbf{C}).$$

**DOWÓD** Zauważmy przede wszystkim, że  $H(\mathbf{K}, \mathbf{P}, \mathbf{C}) = H(\mathbf{C}|\mathbf{K}, \mathbf{P}) + H(\mathbf{K}, \mathbf{P})$ . Klucz i tekst jawny wyznaczają jednoznacznie tekst zaszyfrowany, gdyż  $y = e_K(x)$ . Stąd wynika więc  $H(\mathbf{C}|\mathbf{K}, \mathbf{P}) = 0$ , a dalej także  $H(\mathbf{K}, \mathbf{P}, \mathbf{C}) = H(\mathbf{K}, \mathbf{P})$ . Ponieważ jednak  $\mathbf{K}$  i  $\mathbf{P}$  są niezależne, mamy zatem  $H(\mathbf{K}, \mathbf{P}) = H(\mathbf{K}) + H(\mathbf{P})$ . W konsekwencji

$$H(\mathbf{K}, \mathbf{P}, \mathbf{C}) = H(\mathbf{K}, \mathbf{P}) = H(\mathbf{K}) + H(\mathbf{P}).$$

Podobnie z tego, że klucz i tekst zaszyfrowany wyznaczają jednoznacznie tekst jawny (gdyż  $x = d_K(y)$ ), wnioskujemy, że  $H(\mathbf{P}|\mathbf{K}, \mathbf{C}) = 0$  i w rezultacie  $H(\mathbf{K}, \mathbf{P}, \mathbf{C}) = H(\mathbf{K}, \mathbf{C})$ . Stąd obliczamy, co następuje:

$$\begin{aligned} H(\mathbf{K}|\mathbf{C}) &= H(\mathbf{K}, \mathbf{C}) - H(\mathbf{C}) \\ &= H(\mathbf{K}, \mathbf{P}, \mathbf{C}) - H(\mathbf{C}) \\ &= H(\mathbf{K}) + H(\mathbf{P}) - H(\mathbf{C}), \end{aligned}$$

co daje wymaganą tezę. ■

Powróćmy do przykładu 2.1, by zilustrować ten wynik.

#### Przykład 2.1 cd.

Obliczyliśmy już  $H(\mathbf{P}) \approx 0,81$ ,  $H(\mathbf{K}) = 1,5$  oraz  $H(\mathbf{C}) \approx 1,85$ . Twierdzenie 2.10 mówi nam, że  $H(\mathbf{K}|\mathbf{C}) \approx 1,5 + 0,81 - 1,85 \approx 0,46$ . Możemy to sprawdzić bezpośrednio, stosując definicję entropii warunkowej, w sposób następujący. Najpierw musimy obliczyć prawdopodobieństwa  $p(K_i|j)$ ,  $1 \leq i \leq 3$ ,  $1 \leq j \leq 4$ . Można do tego wykorzystać twierdzenie Bayesa; otrzymamy następujące wyniki:

$$\begin{array}{lll} p(K_1|1) = 1 & p(K_2|1) = 0 & p(K_3|1) = 0 \\ p(K_1|2) = \frac{6}{7} & p(K_2|2) = \frac{1}{7} & p(K_3|2) = 0 \\ p(K_1|3) = 0 & p(K_2|3) = \frac{3}{4} & p(K_3|3) = \frac{1}{4} \\ p(K_1|4) = 0 & p(K_2|4) = 0 & p(K_3|4) = 1 \end{array}$$

I dalej,

$$H(\mathbf{K}|\mathbf{C}) = \frac{1}{8} \cdot 0 + \frac{7}{16} \cdot 0,59 + \frac{1}{4} \cdot 0,81 + \frac{3}{16} \cdot 0 = 0,46,$$

co zgadza się z wynikiem przewidzianym na podstawie twierdzenia 2.10.  $\square$

Załóżmy, że w ramach systemu kryptograficznego ciąg tekstu jawnego

$$x_1 x_2 \dots x_n$$

został za pomocą jednego klucza zaszyfrowany do postaci

$$y_1 y_2 \dots y_n.$$

Przypomnijmy, że podstawowe zadanie kryptoanalytyka polega na ustaleniu klucza. Rozważamy tu ataki oparte tylko na znanym tekście zaszyfrowanym, zakładając, że Oskar dysponuje nieograniczonymi zasobami obliczeniowymi. Przyjmujemy także, że wie on, iż tekst jawnny jest wyrażony w języku „naturalnym”, na przykład po angielsku. Na ogół Oskar będzie w stanie wyeliminować niektóre klucze, pozostanie jednak z wieloma „możliwymi” kluczami, z których tylko jeden jest poprawny. Pozostałe, możliwe, ale niepoprawne, nazwiemy *kluczami błędymi*.

Załóżmy na przykład, że Oskar wszedł w posiadanie tekstu zaszyfrowanego *WNAJW*, otrzymanego przy zastosowaniu szyfru z przesunięciem. Łatwo zauważyc, że są tylko dwa „znaczące” teksty jawne, a mianowicie *river* i *arena*, odpowiadające dwóm możliwym kluczom szyfrowania  $F$  ( $= 5$ ) i  $W$  ( $= 22$ ). Z tych dwóch kluczy jeden będzie kluczem poprawnym, drugi – błędym. (W istocie nie jest bardzo trudno znaleźć ciąg tekstu zaszyfrowanego o długości 5, otrzymanego w wyniku zastosowania szyfru z przesunięciem, któremu można byłoby przypisać dwa różne teksty jawne; Czytelnik może poszukać innych przykładów).

Spróbujmy ustalić ograniczenie górne na oczekiwana liczbę błędnych kluczy. Najpierw musimy określić pojęcie entropii (w przeliczeniu na jedną literę) języka naturalnego  $L$ , którą oznaczymy symbolem  $H_L$ . Entropia  $H_L$  powinna być miarą średniej informacji (na literę) zawartej w „znaczącym” ciągu tekstu jawnego. (Zauważmy, że losowy ciąg znaków alfabetu miałby entropię (na literę) równą  $\log_2 26 \approx 4,70$ ). Pierwszym przybliżeniem  $H_L$  mogłyby być entropia  $H(\mathbf{P})$ . Gdy  $L$  jest językiem angielskim, mamy  $H(\mathbf{P}) \approx 4,19$ , co można wywnioskować z rozkładu prawdopodobieństwa opisanego w tablicy 1.1.

Oczywiście kolejne litery alfabetu nie są niezależne, a korelacje między nimi zmniejszają entropię. Na przykład w języku angielskim po literze „Q” zawsze następuje litera „U”. W ramach przybliżenia „drugiego rzędu” moglibyśmy obliczyć entropię rozkładu prawdopodobieństwa wszystkich digramów, a następnie podzielić ją przez 2. Ogólniej, niech  $\mathbf{P}^n$  będzie zmienną losową, której rozkład pokrywa się z rozkładem prawdopodobieństwa wszystkich  $n$ -gramów tekstu jawnego. Potrzebne nam będą następujące definicje.

**DEFINICJA 2.6**

Niech  $L$  będzie językiem naturalnym. *Entropią* języka  $L$  nazwiemy liczbę  $H_L$ , taką że

$$H_L = \lim_{n \rightarrow \infty} \frac{H(\mathbf{P}^n)}{n},$$

natomiast *nadmiarowość*  $L$  definiujemy jako liczbę  $R_L$ , taką że

$$R_L = 1 - \frac{H_L}{\log_2 |\mathcal{P}|}.$$

**UWAGA**  $H_L$  jest miarą entropii języka  $L$  w przeliczeniu na jedną literę. Język losowy miałby entropię równą  $\log_2 |\mathcal{P}|$ . Tak więc wielkość  $R_L$  określa proporcję „nadmiarowych znaków”, co określamy jako nadmiarowość. ■

W przypadku języka angielskiego przez stablicowanie dużej liczby digramów i częstości ich występowania można oszacować wartość  $H(\mathbf{P}^2)$ . W ten sposób otrzymano wartość przybliżoną  $H(\mathbf{P}^2)/2 \approx 3,9$ . Przez stablicowanie układów trójliterowych itd. można by uzyskać oszacowanie wartości  $H_L$ . W rzeczywistości uzyskano eksperymentalnie wynik empiryczny:  $1 \leq H_L \leq 1,5$ . Oznacza to, że w języku angielskim średnia informacja zawarta w jednej literze mieści się między jednym a półtora bitem na jedną literę!

Jeśli oszacujemy  $H_L$  jako 1,25, otrzymamy nadmiarowość ok. 0,75. To z kolei znaczy, że język angielski jest w 75% nadmiarowy! (Nie należy tego rozumieć tak, że możemy z angielskiego tekstu usunąć trzy z każdych czterech liter i żywić nadzieję, że wciąż będziemy w stanie go odczytać. Oznacza to, że można znaleźć kodowanie Huffmana dla  $n$ -gramów – dla dostatecznie dużej wartości  $n$  – które spowoduje zredukowanie angielskiego tekstu do mniej więcej jednej czwartej jego oryginalnej długości).

Dla danych rozkładów prawdopodobieństwa na  $\mathcal{K}$  i  $\mathcal{P}^n$  możemy określić indekowany przez nie rozkład prawdopodobieństwa na  $\mathcal{C}^n$ , czyli na zbiorze wszystkich  $n$ -gramów tekstu zaszyfrowanego (uzyniliśmy to już w przypadku  $n = 1$ ). Zdefiniowaliśmy wcześniej  $\mathbf{P}^n$  jako zmienną losową reprezentującą  $n$ -gram tekstu jawnego. Podobnie określmy teraz  $\mathbf{C}^n$  jako zmienną losową reprezentującą  $n$ -gram tekstu zaszyfrowanego.

Dla  $y \in \mathbf{C}^n$  niech

$$K(y) = \{K \in \mathcal{K} : \exists \mathbf{x} \in \mathcal{P}^n, p_{\mathcal{P}^n}(\mathbf{x}) > 0, e_K(\mathbf{x}) = y\}.$$

Tak więc  $K(y)$  jest zbiorem tych kluczy  $K$ , dla których  $y$  jest szyfrem pewnego znaczącego ciągu tekstu jawnego o długości  $n$ , czyli zbiorem „możliwych” kluczy prowadzących do tekstu zaszyfrowanego  $y$ . Liczba kluczy błędnych jest wtedy równa  $|K(y)| - 1$ , ponieważ tylko jeden z „możliwych” kluczy jest poprawny. Niech  $\bar{s}_n$  oznacza średnią liczbę błędnych kluczy (względem wszystkich możli-

wych ciągów tekstu zaszyfrowanego o długości  $n$ ). Wartość tę możemy obliczyć następująco

$$\begin{aligned}\bar{s}_n &= \sum_{\mathbf{y} \in \mathcal{C}^n} p(\mathbf{y})(|K(\mathbf{y})| - 1) \\ &= \sum_{\mathbf{y} \in \mathcal{C}^n} p(\mathbf{y})|K(\mathbf{y})| - \sum_{\mathbf{y} \in \mathcal{C}^n} p(\mathbf{y}) \\ &= \sum_{\mathbf{y} \in \mathcal{C}^n} p(\mathbf{y})|K(\mathbf{y})| - 1.\end{aligned}$$

Z twierdzenia 2.10 wiemy, że

$$H(\mathbf{K}|\mathbf{C}^n) = H(\mathbf{K}) + H(\mathbf{P}^n) - H(\mathbf{C}^n).$$

Możemy posłużyć się oszacowaniem

$$H(\mathbf{P}^n) \approx nH_L = n(1 - R_L) \log_2 |\mathcal{P}|,$$

jeśli tylko  $n$  jest dostatecznie duże. Oczywiście,

$$H(\mathbf{C}^n) \leq n \log_2 |\mathcal{C}|.$$

Stąd, jeśli  $|\mathcal{C}| = |\mathcal{P}|$ , to

$$H(\mathbf{K}|\mathbf{C}^n) \geq H(\mathbf{K}) - nR_L \log_2 |\mathcal{P}|. \quad (2.1)$$

Teraz zajmiemy się związkiem wielkości  $H(\mathbf{K}|\mathbf{C}^n)$  z liczbą błędnych kluczy  $\bar{s}_n$ . Obliczamy następująco:

$$\begin{aligned}H(\mathbf{K}|\mathbf{C}^n) &= \sum_{\mathbf{y} \in \mathcal{C}^n} p(\mathbf{y})H(\mathbf{K}|\mathbf{y}) \\ &\leq \sum_{\mathbf{y} \in \mathcal{C}^n} p(\mathbf{y}) \log_2 |K(\mathbf{y})| \\ &\leq \log_2 \sum_{\mathbf{y} \in \mathcal{C}^n} p(\mathbf{y})|K(\mathbf{y})| \\ &= \log_2 (\bar{s}_n + 1).\end{aligned}$$


Użyliśmy tu nierówności Jensena (twierdzenie 2.5) dla funkcji  $f(x) = \log_2(x)$ . Otrzymaliśmy zatem nierówność

$$H(\mathbf{K}|\mathbf{C}^n) \leq \log_2 (\bar{s}_n + 1). \quad (2.2)$$

Łącząc nierówności (2.1) i (2.2), otrzymujemy

$$\log_2 (\bar{s}_n + 1) \geq H(\mathbf{K}) - nR_L \log_2 |\mathcal{P}|.$$

Gdy klucze są wybierane z jednakowym prawdopodobieństwem (co maksymalizuje wartość  $H(\mathbf{K})$ ), mamy następujący wynik:

**TWIERDZENIE 2.11**

Niech  $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$  będzie systemem kryptograficznym, w którym klucze są wybierane z jednakowym prawdopodobieństwem oraz  $|\mathcal{C}| = |\mathcal{P}|$ . Niech ponadto  $R_L$  oznacza nadmiarowość używanego języka. Wówczas, jeśli串tekst zaszyfrowanego ma długość  $n$  dla dostatecznie dużego  $n$ , to oczekiwana liczba błędnych kluczy  $\bar{s}_n$  spełnia następujący warunek:

$$\bar{s}_n \geq \frac{|\mathcal{K}|}{|\mathcal{P}|^{nR_L}} - 1.$$

Przy wzroście  $n$  wartość  $|\mathcal{K}|/|\mathcal{P}|^{nR_L} - 1$  dąży wykładniczo do 0. Zauważmy także, że dla małych wartości  $n$  oszacowanie może nie być dobre, tym bardziej że dla małych wartości  $n$  liczba  $H(\mathbf{P}^n)/n$  może nie być dobrym oszacowaniem dla  $H_L$ .

Musimy zdefiniować jeszcze jedno pojęcie.

**DEFINICJA 2.7**

*Długością krytyczną* (ang. *unicity distance*)<sup>†</sup> systemu kryptograficznego nazywamy tę wartość  $n_0$  zmiennej  $n$ , dla której oczekiwana liczba błędnych kluczy staje się równa 0, czyli średnią długość tekstu zaszyfrowanego, która wystarczy przeciwnikowi do jednoznacznego wyznaczenia klucza, jeśli tylko dysponuje dostatecznie długim czasem obliczeniowym.

Jeśli w twierdzeniu 2.11 przyjmiemy  $\bar{s}_n = 0$  i rozwiążemy nierówność ze wzgledu na  $n$ , otrzymamy oszacowanie długości krytycznej, mianowicie:

$$n_0 \approx \frac{\log_2 |\mathcal{K}|}{R_L \log_2 |\mathcal{P}|}.$$

Rozważmy jako przykład szyfr podstawieniowy. W tym systemie kryptograficznym  $|\mathcal{P}| = 26$  oraz  $|\mathcal{K}| = 26!$ . Jeśli przyjmiemy  $R_L = 0,75$ , otrzymamy następujące oszacowanie długości krytycznej:

$$n_0 \approx 88,4 / (0,75 \cdot 4,7) \approx 25.$$

Ten wynik sugeruje, że do jednoznacznego odtworzenia klucza wystarczy (zazwyczaj)串tekst zaszyfrowanego o długości 25.

## 2.5. Produktowe systemy kryptograficzne

Kolejną innowację wprowadzoną przez Shannona w jego pracy z 1949 roku jest pomysł łączenia systemów kryptograficznych w postaci produktu lub iloczynu

<sup>†</sup>W literaturze można spotkać również określenie *odległość jednostkowa* (przyp. tłum.).

kartezjańskiego. Pomyśl ten nabrał szczególnej wagi przy projektowaniu współczesnych systemów kryptograficznych, takich jak DES, o którym będzie mowa w kolejnym rozdziale.

Dla uproszczenia ograniczymy się tu do badania systemów, w których  $\mathcal{C} = \mathcal{P}$ ; system tego rodzaju nazywa się systemem *endomorficznym*. Niech  $\mathbf{S}_1 = (\mathcal{P}, \mathcal{P}, \mathcal{K}_1, \mathcal{E}_1, \mathcal{D}_1)$  i  $\mathbf{S}_2 = (\mathcal{P}, \mathcal{P}, \mathcal{K}_2, \mathcal{E}_2, \mathcal{D}_2)$  będą dwoma endomorficznymi systemami kryptograficznymi nad tą samą przestrzenią tekstów jawnych (i zaszyfrowanych). Wówczas produktem  $\mathbf{S}_1$  i  $\mathbf{S}_2$ , oznaczonym przez  $\mathbf{S}_1 \times \mathbf{S}_2$ , nazwiemy system kryptograficzny

$$(\mathcal{P}, \mathcal{P}, \mathcal{K}_1 \times \mathcal{K}_2, \mathcal{E}, \mathcal{D}).$$

W systemie produktowym klucz ma postać  $K = (K_1, K_2)$ , gdzie  $K_1 \in \mathcal{K}_1$  i  $K_2 \in \mathcal{K}_2$ . Reguły szyfrowania i deszyfrowania są definiowane następująco: dla każdego  $K = (K_1, K_2)$  mamy regułę szyfrowania  $e_K$  określona wzorem

$$e_{(K_1, K_2)}(x) = e_{K_2}(e_{K_1}(x)),$$

oraz regułę deszyfrowania określona wzorem

$$d_{(K_1, K_2)}(x) = d_{K_1}(d_{K_2}(y)).$$

Inaczej mówiąc, najpierw szyfrujemy  $x$  za pomocą reguły  $e_{K_1}$ , następnie ponownie szyfrujemy otrzymany tekst zaszyfrowany za pomocą reguły  $e_{K_2}$ . Deszyfrowanie odbywa się w podobny sposób, lecz w odwrotnej kolejności:

$$\begin{aligned} d_{(K_1, K_2)}(e_{(K_1, K_2)}(x)) &= d_{(K_1, K_2)}(e_{K_2}(e_{K_1}(x))) \\ &= d_{K_1}(d_{K_2}(e_{K_2}(e_{K_1}(x)))) \\ &= d_{K_1}(e_{K_1}(x)) \\ &= x. \end{aligned}$$

Przypomnijmy, że z przestrzenią kluczy w dowolnym systemie kryptograficznym jest związany pewien rozkład prawdopodobieństwa. Musimy zatem określić także rozkład prawdopodobieństwa dla przestrzeni kluczy  $\mathcal{K}$  systemu produktowego. Zrobimy to w sposób bardzo naturalny:

$$p_{\mathcal{K}}(K_1, K_2) = p_{\mathcal{K}_1}(K_1) \cdot p_{\mathcal{K}_2}(K_2).$$

Tak więc wybieramy  $K_1$  zgodnie z rozkładem  $p_{\mathcal{K}_1}$  i niezależnie wybieramy  $K_2$  zgodnie z rozkładem  $p_{\mathcal{K}_2}$ .

Oto prosty przykład ilustrujący definicję produktowego systemu kryptograficznego. Niech **szyfr multiplikatywny** będzie określony tak jak na rysunku 2.2.

Załóżmy, że  $\mathbf{M}$  jest szyfrem multiplikatywnym (w którym klucze wybierane są z jednakowym prawdopodobieństwem), a  $\mathbf{S}$  jest szyfrem z przesunięciem. Łatwo wówczas zauważyc, że  $\mathbf{M} \times \mathbf{S}$  jest po prostu szyfrem afincznym (także z kluczami wybieranymi z jednakowym prawdopodobieństwem). Nieco trudniej wykazać, że również  $\mathbf{S} \times \mathbf{M}$  jest szyfrem afincznym z jednakowo prawdopodobnymi kluczami.

Niech  $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}$  oraz

$$\mathcal{K} = \{a \in \mathbb{Z}_{26} : \text{NWD}(a, 26) = 1\}.$$

Dla  $a \in \mathcal{K}$  definiujemy

$$e_a(x) = ax \bmod 26$$

oraz

$$d_a(y) = a^{-1}y \bmod 26$$

$$(x, y \in \mathbb{Z}_{26}).$$

### RYSUNEK 2.2. Szyfr mnożnikowy

Udowodnijmy te stwierdzenia. Kluczem w szyfrze z przesunięciem jest element  $K \in \mathbb{Z}_{26}$ , a odpowiadająca mu reguła szyfrowania jest dana równością  $e_K(x) = x + K \bmod 26$ . Z kolei kluczem w szyfrze mnożnikowym jest element  $a \in \mathbb{Z}_{26}$ , taki że  $\text{NWD}(a, 26) = 1$  i wyznaczający regułę szyfrowania  $e_a(x) = ax \bmod 26$ . Stąd klucz w szyfrze produktowym  $\mathbf{M} \times \mathbf{S}$  ma postać  $(a, K)$ , przy czym

$$e_{(a,K)}(x) = ax + K \bmod 26.$$

Tak właśnie jest zdefiniowany klucz w szyfrze afincznym. Dalej, prawdopodobieństwo klucza w szyfrze afincznym jest równe  $1/312 = 1/12 \cdot 1/26$ , co jest iloczynem prawdopodobieństw kluczy  $a$  i  $K$ , odpowiednio. Tak więc  $\mathbf{M} \times \mathbf{S}$  jest szyfrem afincznym.

Rozpatrzmy teraz system  $\mathbf{S} \times \mathbf{M}$ . W tym systemie klucz ma postać  $(K, a)$ , przy czym

$$e_{(K,a)}(x) = a(x + K) = ax + aK \bmod 26.$$

Tak więc klucz  $(K, a)$  szyfru produktowego  $\mathbf{S} \times \mathbf{M}$  jest identyczny z kluczem  $(a, aK)$  szyfru afincznego. Pozostaje wykazać, że każdy klucz szyfru afincznego pojawia się z jednakowym prawdopodobieństwem  $1/312$  w szyfrze produktowym  $\mathbf{S} \times \mathbf{M}$ . Zauważmy, że  $aK = K_1$  wtedy i tylko wtedy, gdy  $K = a^{-1}K_1$  (przypomnijmy, że  $\text{NWD}(a, 26) = 1$ , więc  $a$  ma element odwrotny). Innymi słowy, klucz  $(a, K_1)$  szyfru afincznego jest równoważny kluczowi  $(a^{-1}K_1, a)$  szyfru produktowego  $\mathbf{S} \times \mathbf{M}$ . Mamy zatem bijekcję między dwiema przestrzeniami kluczy. Biorąc pod uwagę, że każdy klucz jest tak samo prawdopodobny, wnioskujemy, że  $\mathbf{S} \times \mathbf{M}$  istotnie jest szyfrem afincznym.

Wykazaliśmy, że  $\mathbf{M} \times \mathbf{S} = \mathbf{S} \times \mathbf{M}$ . Można więc stwierdzić, że dla tych dwóch systemów produktowanie jest *przemienne*. Jednak nie jest tak dla wszystkich par systemów kryptograficznych; nietrudno wskazać kontrprzykłady.

Z drugiej strony, operacja produktowania jest zawsze *łączna*:  $(\mathbf{S}_1 \times \mathbf{S}_2) \times \mathbf{S}_3 = \mathbf{S}_1 \times (\mathbf{S}_2 \times \mathbf{S}_3)$ .

Jeśli weźmiemy produkt endomorficznego systemu kryptograficznego  $\mathbf{S}$  z nim samym, otrzymamy system  $\mathbf{S} \times \mathbf{S}$ , który oznaczmy symbolem  $\mathbf{S}^2$ . Biorąc  $n$ -krotny produkt, otrzymamy system, który oznaczmy symbolem  $\mathbf{S}^n$  i nazwijmy *iterowanym* systemem kryptograficznym.

Mówimy, że system kryptograficzny  $\mathbf{S}$  jest *idempotentny*, gdy  $\mathbf{S}^2 = \mathbf{S}$ . Wiele spośród systemów omawianych w rozdziale 1 to systemy idempotentne, na przykład szyfry: z przesunięciem, podstawieniowy, afiniczy, Hilla, Vigenère'a oraz permutujący. Oczywiście w przypadku systemu idempotentnego  $\mathbf{S}$  nie ma powodu używać systemu produktowego  $\mathbf{S}^2$ , gdyż wymaga on dodatkowego klucza, nie zapewniając większego bezpieczeństwa.

Jeśli system kryptograficzny nie jest idempotentny, to kilkukrotna iteracja potencjalnie zwiększa bezpieczeństwo. Pomyśl ten wykorzystuje się w standardzie szyfrowania danych, który składa się z 16 iteracji. Do tego jest jednak potrzebny niedempotentny wyjściowy system kryptograficzny. Jednym ze sposobów na zbudowanie niedempotentnego prostego systemu jest wzięcie produktu dwóch różnych (prostych) systemów.

**UWAGA** Nietrudno wykazać, że jeśli oba systemy  $\mathbf{S}_1$  i  $\mathbf{S}_2$  są idempotentne i przemienne, to system  $\mathbf{S}_1 \times \mathbf{S}_2$  także jest idempotentny. Wynika to z następujących obliczeń:

$$\begin{aligned} (\mathbf{S}_1 \times \mathbf{S}_2) \times (\mathbf{S}_1 \times \mathbf{S}_2) &= \mathbf{S}_1 \times (\mathbf{S}_2 \times \mathbf{S}_1) \times \mathbf{S}_2 \\ &= \mathbf{S}_1 \times (\mathbf{S}_1 \times \mathbf{S}_2) \times \mathbf{S}_2 \\ &= (\mathbf{S}_1 \times \mathbf{S}_1) \times (\mathbf{S}_2 \times \mathbf{S}_2) \\ &= (\mathbf{S}_1 \times \mathbf{S}_2) \end{aligned}$$

(Warto zwrócić uwagę na wykorzystanie łączności w tym dowodzie).

Tak więc, jeśli oba systemy  $\mathbf{S}_1$  i  $\mathbf{S}_2$  są idempotentne, a chcemy, by system  $\mathbf{S}_1 \times \mathbf{S}_2$  idempotentny nie był, musimy się upewnić, że produktowanie tych systemów nie jest przemienne. ■

Na szczęście wiele prostych systemów kryptograficznych można przy tym podejściu wykorzystać jako proste elementy konstrukcyjne. Często stosuje się technikę łączenia w ten sposób szyfrów typu podstawieniowego z szyframi typu permutującego. W następnym rozdziale zobaczymy, jak to wygląda w realizacji.

## 2.6. Uwagi

Pionierem w użyciu w kryptografi metod opartych na entropii oraz pojęcia tajności doskonałej był Shannon [SH49]. W tej samej pracy opisał on też systemy produktowe. Samo pojęcie entropii także pochodzi od Shannona [SH48]. Dobre

wprowadzenie do entropii, kodowań Huffmana oraz innych pokrewnych tematów można znaleźć w książkach Welsza [WE88] oraz Goldiego i Pincha [GP91].

Wyniki przytaczane w podrozdziale 2.4 zawsze możemy Beaucheminowi i Brasardowi [BB88], którzy uogólnili wcześniejsze wyniki Shannona.

## Ćwiczenia

- 2.1. Niech  $n$  będzie dodatnią liczbą całkowitą. *Kwadratem łacińskim* rzędu  $n$  nazywa się macierz  $L$  o  $n$  wierszach i  $n$  kolumnach, której elementami są liczby całkowite  $1, \dots, n$  rozmieszczone tak, że każda z nich występuje dokładnie raz w każdym wierszu i każdej kolumnie. Oto przykład kwadratu łacińskiego rzędu 3:

1	2	3
3	1	2
2	3	1

Dla danego kwadratu łacińskiego  $L$  rzędu  $n$  możemy zdefiniować pewien związany z nim system kryptograficzny. Przyjmijmy  $\mathcal{P} = \mathcal{C} = \mathcal{K} = \{1, \dots, n\}$ . Dla  $1 \leq i \leq n$  niech reguła szyfrowania  $e_i$  będzie określona następująco:  $e_i(j) = L(i, j)$ . (Każdy wiersz kwadratu generuje w ten sposób jedną regułę szyfrowania).

Podaj pełny dowód tego, że system kryptograficzny wykorzystujący kwadrat łaciński zapewnia tajność doskonałą.

- 2.2. Wykaż, że szyfr afiniczny zapewnia tajność doskonałą.
- 2.3. Załóżmy, że pewien system kryptograficzny zapewnia tajność doskonałą dla pewnego rozkładu prawdopodobieństwa tekstu jawnego. Wykaż, że zachowuje tajność doskonałą dla *dowolnego* rozkładu prawdopodobieństwa tekstu jawnego.
- 2.4. Wykaż, że jeśli system kryptograficzny zapewnia tajność doskonałą oraz  $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{P}|$ , to każdy tekst zaszyfrowany jest równie prawdopodobny.
- 2.5. Niech  $\mathbf{X}$  będzie zbiorem mocy  $n$  dla pewnego  $n$ , takiego że  $2^k \leq n \leq 2^{k+1}$ , oraz  $p(x) = 1/n$  dla wszystkich  $x \in \mathbf{X}$ .

- (a) Znajdź wolne od przedrostków kodowanie  $f$  zbioru  $\mathbf{X}$ , takie że  $\ell(f) = k + 2 - 2^{k+1}/n$ .

**WSKAZÓWKA** Zakoduj  $2^{k+1} - n$  elementów zbioru  $\mathbf{X}$  w postaci ciągów o długości  $k$ , a pozostałe elementy jako ciągi długości  $k+1$ .

- (b) Przedstaw na przykładzie swoją konstrukcję, gdy  $n = 6$ . Oblicz w tym przypadku  $\ell(f)$  oraz  $H(\mathbf{X})$ .

- 2.6. Załóżmy, że  $\mathbf{X} = \{a, b, c, d, e\}$  ma następujący rozkład prawdopodobieństwa:  $p(a) = 0,32$ ,  $p(b) = 0,23$ ,  $p(c) = 0,20$ ,  $p(d) = 0,15$  oraz  $p(e) = 0,10$ . Korzystając z algorytmu Huffmana, znajdź optymalne kodowanie  $\mathbf{X}$  wolne od przedrostków. Porównaj długość tego kodowania z wartością  $H(\mathbf{X})$ .

- 2.7. Udowodnij, że  $H(\mathbf{X}, \mathbf{Y}) = H(\mathbf{Y}) + H(\mathbf{X}|\mathbf{Y})$ . Następnie wywnioskuj stąd, że  $H(\mathbf{X}|\mathbf{Y}) \leq H(\mathbf{X})$ , przy czym równość ma miejsce wtedy i tylko wtedy, gdy  $\mathbf{X}$  i  $\mathbf{Y}$  są niezależne.
- 2.8. Wykaż, że system kryptograficzny zapewnia tajność doskonałą wtedy i tylko wtedy, gdy  $H(\mathbf{P}|\mathbf{C}) = H(\mathbf{P})$ .
- 2.9. Udowodnij, że w dowolnym systemie kryptograficznym  $H(\mathbf{K}|\mathbf{C}) \geq H(\mathbf{P}|\mathbf{C})$ . (Intuicyjnie, z nierówności tej wynika, że gdy przeciwnik ma w ręku tekst zaszyfrowany, jego niepewność co do klucza jest co najmniej taka, jak jego niepewność co do tekstu jawnego).
- 2.10. Rozważmy system kryptograficzny, w którym  $\mathcal{P} = \{a, b, c\}$ ,  $\mathcal{K} = \{K_1, K_2, K_3\}$  oraz  $\mathcal{C} = \{1, 2, 3, 4\}$ . Przypuśćmy, że macierz szyfrowania ma postać następującą:

	$a$	$b$	$c$
$K_1$	1	2	3
$K_2$	2	3	4
$K_3$	3	4	1

Przyjmując, że klucze wybierane są z jednakowym prawdopodobieństwem, a rozkład prawdopodobieństwa tekstu jawnego jest następujący:  $p_{\mathcal{P}}(a) = 1/2$ ,  $p_{\mathcal{P}}(b) = 1/3$ ,  $p_{\mathcal{P}}(c) = 1/6$ , oblicz  $H(\mathbf{P})$ ,  $H(\mathbf{C})$ ,  $H(\mathbf{K})$ ,  $H(\mathbf{K}|\mathbf{C})$  oraz  $H(\mathbf{P}|\mathbf{C})$ .

- 2.11. Oblicz wartości  $H(\mathbf{K}|\mathbf{C})$  i  $H(\mathbf{K}|\mathbf{P}, \mathbf{C})$  dla szyfru afinielnego.
- 2.12. Rozważmy szyfr Vigenèra'a ze słowem kluczowym o długości  $m$ . Wykaż, że długość krytyczna jest równa  $1/R_L$ , gdzie  $R_L$  jest nadmiarowością używanego języka. (Ten wynik można interpretować w sposób następujący. Jeśli  $n_0$  oznacza liczbę szyfrowanych znaków alfabetu, to „długość” tekstu jawnego wynosi  $n_0/m$ , ponieważ każdy element tekstu jawnego składa się z  $m$  znaków. Tak więc wartość  $1/R_L$ , jaką przyjmuje długość krytyczna, odpowiada tekstowi jawnemu, złożonemu z  $m/R_L$  znaków alfabetu).
- 2.13. Wykaż, że długość krytyczna szyfru Hilla (z macierzą szyfrowania  $m \times m$ ) jest mniejsza od  $m/R_L$ . (Zauważ, że liczba znaków alfabetu w tekście o tej długości jest równa  $m^2/R_L$ ).
- 2.14. W szyfrze podstawieniowym nad  $n$ -elementową przestrzenią tekstu jawnego mamy  $|\mathcal{K}| = n!$ . Wzór Stirlinga daje następujące oszacowanie liczby  $n!$ :
- $$n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n.$$
- (a) Korzystając ze wzoru Stirlinga, znajdź oszacowanie długości krytycznej szyfru podstawieniowego.
- (b) Niech  $m \geq 1$  będzie liczbą całkowitą. Szyfr podstawieniowy będziemy nazywać  **$m$ -gramowym**, gdy przestrzeń tekstów jawnych (i tekstów zaszyfrowanych) składa się ze wszystkich  $26^m$   $m$ -gramów (ciągów  $m$ -literowych). Oszacuj długość krytyczną  $m$ -gramowego szyfru podstawieniowego, gdy  $R_L = 0,75$ .
- 2.15. Wykaż, że szyfr z przesunięciem jest idempotentny.

- 2.16. Niech  $\mathbf{S}_1$  będzie szyfrem z przesunięciem (z równoprawdopodobnymi kluczami, jak zwykle), a  $\mathbf{S}_2$  szyfrem z przesunięciem, w którym klucze wybierane są zgodnie z pewnym rozkładem prawdopodobieństwa  $p_{\mathcal{K}}$  (który nie musi zapewniać jednakoowego prawdopodobieństwa). Wykaż, że  $\mathbf{S}_1 \times \mathbf{S}_2 = \mathbf{S}_1$ .
- 2.17. Niech  $\mathbf{S}_1$  i  $\mathbf{S}_2$  będą szyframi Vigenèr'a ze słowami kluczowymi o długościach  $m_1, m_2$  odpowiednio, przy czym  $m_1 > m_2$ .
- (a) Wykaż, że jeśli  $m_2|m_1$ , to  $\mathbf{S}_2 \times \mathbf{S}_1 = \mathbf{S}_1$ .
  - (b) Można by przypuszczać, że zachodzi wynik ogólniejszy: jeśli  $\mathbf{S}_3$  jest szyfrem Vigenèr'a ze słowem kluczowym o długości  $\text{NWW}(m_1, m_2)$ , to  $\mathbf{S}_2 \times \mathbf{S}_1 = \mathbf{S}_3$ . Udowodnij, że to przypuszczenie jest błędne.
- WSKAZÓWKA** Jeśli  $m_1 \not\equiv 0 \pmod{m_2}$ , to liczba kluczy w produktowym systemie kryptograficznym  $\mathbf{S}_2 \times \mathbf{S}_1$  jest mniejsza niż liczba kluczy w  $\mathbf{S}_3$ .

# 3

---

## Standard szyfrowania danych – DES

---

### 3.1. Wprowadzenie

Dnia 15 maja 1973 roku National Bureau of Standards (Narodowe Biuro Standardyzacji) ogłosiło w Rejestrze Federalnym zamówienie publiczne na systemy kryptograficzne. Tak rozpoczął się proces opracowywania **standardu szyfrowania danych** (ang. *Data Encryption Standard*, DES), który stał się najpowszechniej stosowanym w świecie systemem kryptograficznym. DES powstał w firmie IBM jako modyfikacja wcześniejszego systemu, znanego pod nazwą **LUCIFER**. Opublikowano go po raz pierwszy w Rejestrze Federalnym 17 marca 1975 roku. Po długiej, publicznej dyskusji DES został 15 stycznia 1977 roku przyjęty jako standard do „nietajnych” zastosowań. Od momentu jego przyjęcia, mniej więcej co pięć lat, DES jest poddawany rewizji przez Narodowe Biuro Standardyzacji. Ostatnie przedłużenie ważności miało miejsce w styczniu 1994 roku; jego ważność upłynie w 1998 roku. Przewiduje się, że po tej dacie DES straci swój status standardu<sup>†</sup>.

---

### 3.2. Opis DES

Pełny opis systemu DES znajduje się w publikacji nr 46 Federal Information Processing Standard (Federalnego Standardu Przetwarzania Informacji), datowanej 15 stycznia 1977 roku. W systemie DES szyfruje się 64-bitowy ciąg tekstu jawnego  $x$  za pomocą klucza  $K$ , będącego ciągiem bitów o długości 56, i otrzymuje w wyniku 64-bitowy ciąg tekstu zaszyfrowanego. Przedstawimy najpierw ogólny opis tego systemu.

Algorytm jest realizowany w trzech etapach.

- (1) Dla danego tekstu jawnego  $x$  jest tworzony ciąg bitów  $x_0$ , przez wykonanie permutacji bitów ciągu  $x$  zgodnie z pewną (ustaloną) permutacją

---

<sup>†</sup>Angielski oryginał został wydany w 1995 roku (przyp. tłum.).

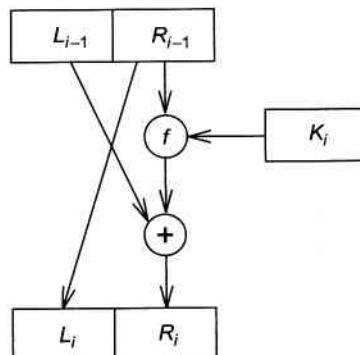
początkową IP. Piszemy:  $x_0 = \text{IP}(x) = L_0R_0$ , gdzie  $L_0$  oznacza pierwsze 32 bity ciągu  $x_0$ ,  $R_0$  zaś to ostatnie 32 bity ciągu  $x_0$ .

- (2) Następnie jest obliczanych 16 iteracji pewnej funkcji. Dla  $1 \leq i \leq 16$  wyznacza się  $L_iR_i$  zgodnie z następującą regułą:

$$\begin{aligned} &L_iR_{i-1} \\ &R_iL_{i-1} \oplus f(R_{i-1}, K_i), \end{aligned}$$

gdzie  $\oplus$  oznacza operację różnicę symetryczną określoną na dwóch ciągach bitów. Funkcję  $f$  opiszymy później, natomiast  $K_1, K_2, \dots, K_{16}$  (tworzące zestaw podkluczów (ang. *key schedule*)) są ciągami 48 bitów obliczanych jako funkcje klucza  $K$ . (W rzeczywistości  $K_i$  jest pewnym permutowanym wyborem bitów klucza  $K$ ). Na rysunku 3.1 jest przedstawiona jedna runda procesu szyfrowania.

- (3) Wykonując permutację  $\text{IP}^{-1}$ , odwrotną do permutacji początkowej, do ciągu  $R_{16}L_{16}$ , otrzymuje się tekst zaszyfrowany  $y$ . Tak więc  $y = \text{IP}^{-1}(R_{16}L_{16})$ . Warto tu zwrócić uwagę na odwróconą kolejność  $L_{16}$  i  $R_{16}$ .



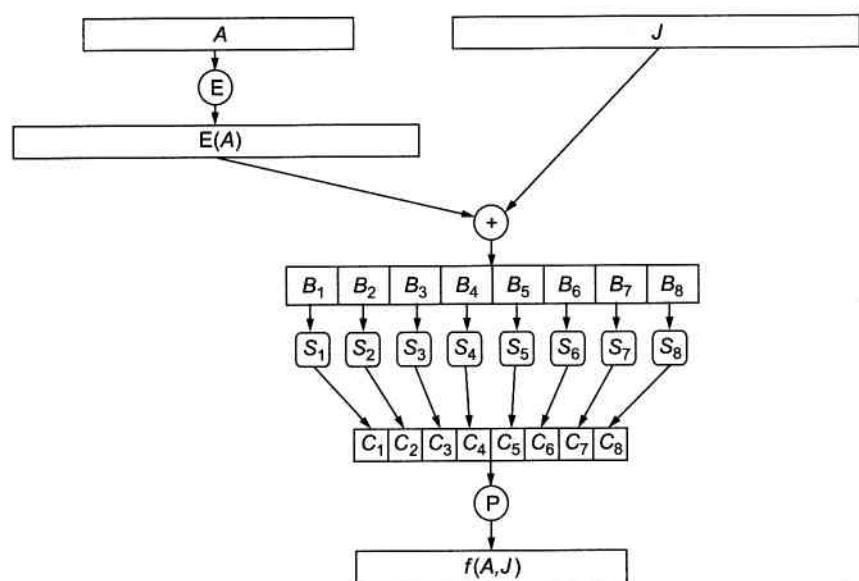
**RYSUNEK 3.1.** Jedna runda szyfrowania w systemie DES

Wartość funkcji  $f$  (ciąg 32-bitowy) zależy od dwóch argumentów: ciągu  $A$  o długości 32 bitów oraz ciągu  $J$  o długości 48 bitów. Wartość ta jest obliczana w następujący sposób.

- (1) Pierwszy argument  $A$  jest „rozszerzany” do 48 bitów zgodnie z pewną funkcją rozszerzania E. E( $A$ ) składa się z 32 bitów ciągu  $A$ , poddanych pewnej permutacji, oraz z powtórnych wystąpień 16 spośród nich.
- (2) Następnie jest obliczana różnica symetryczna  $E(A) \oplus J$ , a wynik jest zapisywany w postaci konkatenacji ośmiu 6-bitowych ciągów:  $B = B_1B_2B_3B_4B_5B_6B_7B_8$ .

- (3) W następnym kroku jest używanych osiem bloków podstawień zwanych *S-blokami* (ang. *S-boxes*)  $S_1, \dots, S_8$ . Każdy blok  $S_i$  jest ustaloną macierzą  $4 \times 16$  o elementach ze zbioru  $\{0, 1, \dots, 15\}$ . Mając dany ciąg bitów o długości 6, na przykład  $B_j = b_1 b_2 b_3 b_4 b_5 b_6$ , oblicza się  $S_j(B_j)$  w sposób następujący. Dwa krańcowe bity  $b_1 b_6$  określają binarną reprezentację wiersza  $r$  macierzy  $S_j$  ( $0 \leq r \leq 3$ ), a cztery bity  $b_2 b_3 b_4 b_5$  są binarną reprezentacją kolumny  $c$  macierzy  $S_j$  ( $0 \leq c \leq 15$ ). Wtedy  $S_j(B_j)$  jest definiowany jako element  $S_j(r, c)$ , zapisany binarnie w postaci ciągu bitów o długości 4. (Możemy zatem traktować macierz  $S_j$  jako funkcję, której argumentami są pary ciągów – jeden 2-bitowy, drugi 4-bitowy, wynikiem zaś ciąg 4-bitowy). W ten sposób otrzymujemy  $C_j = S_j(B_j)$ ,  $1 \leq j \leq 8$ .

(4) Na ciągu  $C = C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8$  o długości 32 bitów wykonuje się permutację  $P$ . Otrzymany w ten sposób ciąg  $P(C)$  jest wartością  $f(A, J)$ .



### RYSUNEK 3.2. Funkcja $f$ systemu DES

Funkcja  $f$  jest zilustrowana na rysunku 3.2. Zasadniczo składa się na nią podstawienie (z użyciem bloku podstawienia), po którym stosuje się (ustaloną) permutację  $P$ . Wspomniane 16 iteracji funkcji  $f$  składają się na system produktywy, opisany w podrozdziale 2.5.

W pozostałej części tego podrozdziału opiszymy funkcje stosowane w systemie DES.

Permutacja początkowa IP ma następującą postać:

IP									
58	50	42	34	26	18	10	2		
60	52	44	36	28	20	12	4		
62	54	46	38	30	22	14	6		
64	56	48	40	32	24	16	8		
57	49	41	33	25	17	9	1		
59	51	43	35	27	19	11	3		
61	53	45	37	29	21	13	5		
63	55	47	39	31	23	15	7		

Oznacza to, że 58 bit ciągu  $x$  jest pierwszym bitem w  $\text{IP}(x)$ , 50 bit w  $x$  jest drugim bitem w  $\text{IP}(x)$  itd.

Permutację odwrotną  $\text{IP}^{-1}$  opisuje poniższa tablica:

$\text{IP}^{-1}$								
40	8	48	16	56	24	64	32	
39	7	47	15	55	23	63	31	
38	6	46	14	54	22	62	30	
37	5	45	13	53	21	61	29	
36	4	44	12	52	20	60	28	
35	3	43	11	51	19	59	27	
34	2	42	10	50	18	58	26	
33	1	41	9	49	17	57	25	

Kolejna tablica określa funkcję rozszerzenia E.

Tabela wyboru bitów funkcji E					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Oto kolejno, osiem bloków podstawień i permutacja P.

$S_1$															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

$S_2$															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

$S_3$															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

$S_4$															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

$S_5$															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

$S_6$															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

$S_7$															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

$S_8$															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Na koniec musimy jeszcze opisać proces obliczania zestawu podkluczów na podstawie klucza  $K$ . Klucz  $K$  jest w rzeczywistości ciągiem 64 bitów, z których 56 stanowi klucz właściwy, a 8 pozostałych to bity parzystości (do kontroli błędów). Bity na pozycjach 8, 16, ..., 64 są określone tak, by każdy bajt składał się z nieparzystej liczby jedynek. Tak więc można wychwycić błąd w każdej 8-bitowej grupie. Podczas obliczania zestawu podkluczów bity parzystości są pomijane.

- (1) Bity parzystości, występujące w 64-bitowym kluczu  $K$ , są odrzucane, pozostałe bity podlegają (ustalonej) permutacji PC-1. Będziemy pisali  $\text{PC-1}(K) = C_0 D_0$ , gdzie  $C_0$  obejmuje pierwszych 28 bitów ciągu  $\text{PC-1}(K)$ ,  $D_0$  obejmuje ostatnich 28 bitów.
- (2) Dla  $i$  od 1 do 16 oblicza się

$$C_i = LS_i(C_{i-1})$$

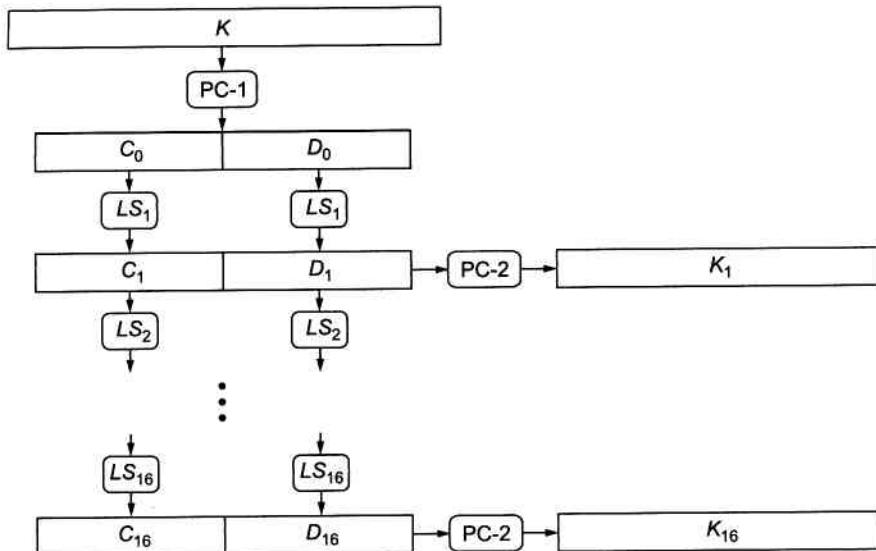
$$D_i = LS_i(D_{i-1})$$

oraz  $K_i = \text{PC-2}(C_i D_i)$ .  $LS_i$  reprezentuje cykliczne przesunięcie (w lewo) o jedną lub dwie pozycje, w zależności od wartości  $i$ : o jedną pozycję, gdy  $i = 1, 2, 9$  lub  $16$ , o dwie pozycje w przeciwnym przypadku. PC-2 jest inną ustaloną permutacją.

Obliczenie zestawu podkluczów ilustruje rysunek 3.3.

Permutacje PC-1 i PC-2, stosowane przy obliczaniu zestawu podkluczów, są określone w sposób następujący.

PC-1							
57	49	41	33	25	17	9	
1	58	50	42	34	26	18	
10	2	59	51	43	35	27	
19	11	3	60	52	44	36	
63	55	47	39	31	23	15	
7	62	54	46	38	30	22	
14	6	61	53	45	37	29	
21	13	5	28	20	12	4	



RYSUNEK 3.3. Obliczanie zestawu podkluczy w systemie DES

PC-2					
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Oto zestaw kluczy otrzymany w wyniku takiego obliczenia. Jak stwierdziliśmy wyżej, w każdej rundzie stosuje się klucz 48-bitowy, złożony z bitów pochodzących z klucza  $K$ . Elementy tablic przedstawionych poniżej odnoszą się do bitów klucza  $K$  użytych w danej rundzie.

Runda 1											
10	51	34	60	49	17	33	57	2	9	19	42
3	35	26	25	44	58	59	1	36	27	18	41
22	28	39	54	37	4	47	30	5	53	23	29
61	21	38	63	15	20	45	14	13	62	55	31

Runda 2											
2	43	26	52	41	9	25	49	59	1	11	34
60	27	18	17	36	50	51	58	57	19	10	33
14	20	31	46	29	63	39	22	28	45	15	21
53	13	30	55	7	12	37	6	5	54	47	23

Runda 3												
51	27	10	36	25	58	9	33	43	50	60	18	
44	11	2	1	49	34	35	42	41	3	59	17	
61	4	15	30	13	47	23	6	12	29	62	5	
37	28	14	39	54	63	21	53	20	38	31	7	

Runda 4												
35	11	59	49	9	42	58	17	27	34	44	2	
57	60	51	50	33	18	19	26	25	52	43	1	
45	55	62	14	28	31	7	53	63	13	46	20	
21	12	61	23	38	47	5	37	4	22	15	54	

Runda 5												
19	60	43	33	58	26	42	1	11	18	57	51	
41	44	35	34	17	2	3	10	9	36	27	50	
29	39	46	61	12	15	54	37	47	28	30	4	
5	63	45	7	22	31	20	21	55	6	62	38	

Runda 6												
3	44	27	17	42	10	26	50	60	2	41	35	
25	57	19	18	1	51	52	59	58	49	11	34	
13	23	30	45	63	62	38	21	31	12	14	55	
20	47	29	54	6	15	4	5	39	53	46	22	

Runda 7												
52	57	11	1	26	59	10	34	44	51	25	19	
9	41	3	2	50	35	36	43	42	33	60	18	
28	7	14	29	47	46	22	5	15	63	61	39	
4	31	13	38	53	62	55	20	23	37	30	6	

Runda 8												
36	41	60	50	10	43	59	18	57	35	9	3	
58	25	52	51	34	19	49	27	26	17	44	2	
12	54	61	13	31	30	6	20	62	47	45	23	
55	15	28	22	37	46	39	4	7	21	14	53	

Runda 9												
57	33	52	42	2	35	51	10	49	27	1	60	
50	17	44	43	26	11	41	19	18	9	36	59	
4	46	53	5	23	22	61	12	54	39	37	15	
47	7	20	14	29	38	31	63	62	13	6	45	

Runda 10													
41	17	36	26	51	19	35	59	33	11	50	44		
34	1	57	27	10	60	25	3	2	58	49	43		
55	30	37	20	7	6	45	63	38	23	21	62		
31	54	4	61	13	22	15	47	46	28	53	29		

Runda 11													
25	1	49	10	35	3	19	43	17	60	34	57		
18	50	41	11	59	44	9	52	51	42	33	27		
39	14	21	4	54	53	29	47	22	7	5	46		
15	38	55	45	28	6	62	31	30	12	37	13		

Runda 12													
9	50	33	59	19	52	3	27	1	44	18	41		
2	34	25	60	43	57	58	36	35	26	17	11		
23	61	5	55	38	37	13	31	6	54	20	30		
62	22	39	29	12	53	46	15	14	63	21	28		

Runda 13													
58	34	17	43	3	36	52	11	50	57	2	25		
51	18	9	44	27	41	42	49	19	10	1	60		
7	45	20	39	22	21	28	15	53	38	4	14		
46	6	23	13	63	37	30	62	61	47	5	12		

Runda 14													
42	18	1	27	52	49	36	60	34	41	51	9		
35	2	58	57	11	25	26	33	3	59	50	44		
54	29	4	23	6	5	12	62	37	22	55	61		
30	53	7	28	47	21	14	46	45	31	20	63		

Runda 15													
26	2	50	11	36	33	49	44	18	25	35	58		
19	51	42	41	60	9	10	17	52	43	34	57		
38	13	55	7	53	20	63	46	21	6	39	45		
14	37	54	12	31	5	61	30	29	15	4	47		

Runda 16													
18	59	42	3	57	25	41	36	10	17	27	50		
11	43	34	33	52	1	2	9	44	35	26	49		
30	5	47	62	45	12	55	38	13	61	31	37		
6	29	46	4	23	28	53	22	21	7	63	39		

Do deszyfrowania wykorzystuje się ten sam algorytm co do szyfrowania, z ciągiem wejściowym  $y$ , lecz zestawu podkluczy używa się w odwrotnej kolejności:  $K_{16}, \dots, K_1$ . Wynikiem jest tekst jawnny  $x$ .

### 3.2.1. Przykład szyfrowania w systemie DES

Oto przykład szyfrowania za pomocą DES. Przypuśćmy, że mamy zaszyfrować tekst jawnego (zapisany w systemie heksadecymalnym<sup>†</sup>)

0123456789ABCDEF

za pomocą (heksadecymalnego) klucza

133457799BBCDFF1.

W systemie binarnym klucz – bez bitów parzystości – wygląda tak:

0001001001101001010110111100100110110111101101111111000.

W wyniku zastosowania permutacji IP otrzymujemy  $L_0$  i  $R_0$  (w postaci binarnej):

$L_0 = 1100110000000000110011001111111$
$L_1 = R_0 = 11110000101010101111000010101010$

Dalej wykonuje się 16 rund szyfrowania, jak poniżej.

$E(R_0) = 01111010000101010101011110100001010101010101$
$K_1 = 0001101100000101110111111100011100001110010$
$E(R_0) \oplus K_1 = 011000010001011110111010100001100110010100100111$
Wyniki S-bloków      01011100100000101011010110010111
$f(R_0, K_1) = 00100011010010101010011011011$
$L_2 = R_1 = 11101111010010100110010101000100$
$E(R_1) = 011101011110101001010100001100001010101000001001$
$K_2 = 011110011010111011011001110110111100100111100101$
$E(R_1) \oplus K_2 = 000011000100010010001101111010110110001111101100$
Wyniki S-bloków      11111000110100000011101010101110
$f(R_1, K_2) = 0011110010101011000011110100011$
$L_3 = R_2 = 110011000000000010111011100001001$
$E(R_2) = 1110010110000000000000001010111010111010000101001$
$K_3 = 010101011111110010001010010000101100111110011001$
$E(R_2) \oplus K_3 = 101100000111110010001000111110000010011111001010$
Wyniki S-bloków      00100111000100001110000101101111
$f(R_2, K_3) = 01001101000101100110111010110000$
$L_4 = R_3 = 10100010010111000000101111110100$
$E(R_3) = 010100000100001011111000000000101011111110101001$
$K_4 = 011100101010110110101101101100110100011101$
$E(R_3) \oplus K_4 = 00100010111011110010110110111100100101010100$
Wyniki S-bloków      00100001111011011001111100111010
$f(R_3, K_4) = 10111011001000110111011101001100$
$L_5 = R_4 = 011101110010001000000000001000101$

<sup>†</sup>Inaczej szesnastkowym (przyp. tłum.).

$E(R_4)$	=	1011101011101001000001000000000000000001000001010
$K_5$	=	0111110011101100000011111010110101001110101000
$E(R_4) \oplus K_5$	=	11000110000001010000001111010110101000110100010
Wyniki $S$ -bloków		01010000110010000011000111101011
$f(R_4, K_5)$	=	00101000000100111010110111000011
$L_6 = R_5$	=	1000101001001111010011000110111

$E(R_5)$	=	11000101010000100101111110100001100000110101111
$K_6$	=	01100011101001010011110010100000111101100101111
$E(R_5) \oplus K_6$	=	101001101110011101100001100000001011101010000000
Wyniki $S$ -bloków		01000001111100110100110000111101
$f(R_5, K_6)$	=	10011110010001011100110100101100
$L_7 = R_6$	=	11101001011001111100110101101001

$E(R_6)$	=	11110101001010110000111111001011010101101010011
$K_7$	=	11101100100001001011011111101100001100010111100
$E(R_6) \oplus K_7$	=	0001100110101111101110000001001110110011111011111
Wyniki $S$ -bloków		00010000011101010100000010101101
$f(R_6, K_7)$	=	10001100000001010001110000100111
$L_8 = R_7$	=	00000110010010101011101000010000

$E(R_7)$	=	00000000110000100101010101011110100000010100000
$K_8$	=	11110111100010100011101011000001001110111111011
$E(R_7) \oplus K_8$	=	11110111010010000110111110011110011110101011011
Wyniki $S$ -bloków		0110110000011000011110010101110
$f(R_7, K_8)$	=	00111100000011101000011011111001
$L_9 = R_8$	=	11010101011010010100101110010000

$E(R_8)$	$=$	011010101010101101010010101001010111110010100001
$K_9$	$=$	11100000110110111110101111011011110011110000001
$E(R_8) \oplus K_9$	$=$	1000101001110000101110010100100010011011001000000
Wyniki $S$ -bloków		00010001000011000101011101110111
$f(R_8, K_9)$	$=$	00100010001101100111110001101010
$L_{10} = R_9$	$=$	00100100011111001100011001111010

$E(R_9)$	$=$	00010000100000111111001011000001100001111110100
$K_{10}$	$=$	101100011111001101000111101110100100011001001111
$E(R_9) \oplus K_{10}$	$=$	10100001011100001011110110110101000010110111011
Wyniki $S$ -bloków		11011010000001000101001001110101
$f(R_9, K_{10})$	$=$	01100010101111001001110000100010
$L_{11} = R_{10}$	$=$	10110111110101011101011110110010

$$\begin{aligned}
 E(R_{10}) &= 010110101111110101010111110101011111101101001 \\
 K_{11} &= 00100001010111111010011110111101101001110000110 \\
 E(R_{10}) \oplus K_{11} &= 011110111010000101111000001101000010111000100011 \\
 \text{Wyniki } S\text{-bloków} & 01110011000001011101000100000001 \\
 f(R_{10}, K_{11}) &= 11100001000001001111101000000010 \\
 L_{12} \equiv B_{11} &= 11000101011110000011110001111000
 \end{aligned}$$

$E(R_{11})$	= 011000001010101111100000001111100000111110001
$K_{12}$	= 0111010101110001111010110010100011001111101001
$E(R_{11}) \oplus K_{12}$	= 000101011101101000001011000101111001000011000
Wyniki S-bloków	0111011100010110010011000110101
$f(R_{11}, K_{12})$	= 110000100110100011001111101010
$L_{13} = R_{12}$	= 01110101101111010001100001011000
$E(R_{12})$	= 00111010101111011111010100011110000001011110000
$K_{13}$	= 10010111110001011101000111110101011101001000001
$E(R_{12}) \oplus K_{13}$	= 101011010111100000101011011101011011100010110001
Wyniki S-bloków	10011010110100011000101101001111
$f(R_{12}, K_{13})$	= 11011101101110110010100100010
$L_{14} = R_{13}$	= 00011000110000110001010101011010
$E(R_{13})$	= 000011110001011000000110100010101010101011110100
$K_{14}$	= 01011111010000111011011111100101110011100111010
$E(R_{13}) \oplus K_{14}$	= 010100000101010110110001011110000100110111001110
Wyniki S-bloków	01100100011110011001101011110001
$f(R_{13}, K_{14})$	= 10110111001100011000111001010101
$L_{15} = R_{14}$	= 11000010100011001001011000001101
$E(R_{14})$	= 11100000010101000101100101001010110000001011011
$K_{15}$	= 101111111001000110001101001111010011111100001010
$E(R_{14}) \oplus K_{15}$	= 01011111110001011101010001110111111111101010001
Wyniki S-bloków	10110010111010001000110100111100
$f(R_{14}, K_{15})$	= 01011011100000010010011101101110
$L_{16} = R_{15}$	= 01000011010000100011001000110100
$E(R_{15})$	= 00100000011010100000100000110100100000110101000
$K_{16}$	= 11001011001111011000101100001110000101111110101
$E(R_{15}) \oplus K_{16}$	= 11101011010101110001111000101000101011001011101
Wyniki S-bloków	10100111100000110010010000101001
$f(R_{15}, K_{16})$	= 11001000110000000100111110011000
$R_{16}$	= 00001010010011001101100110010101

Na koniec, stosując permutację  $IP^{-1}$ , otrzymujemy tekst zaszyfrowany, którym (w postaci heksadecymalnej) jest

85E813540F0AB405.

### 3.3. Kontrowersje wokół DES

Propozycja nadania systemowi DES charakteru standardu spotkała się z rozległą krytyką. Jedno z zastrzeżeń dotyczyło S-bloków. Wszystkie obliczenia przeprowadzane w DES – z wyjątkiem S-bloków – są *liniowe*. Na przykład obliczenie różnicy symetrycznej dwóch wartości wyjściowych daje ten sam wynik, co

obliczenie różnicy symetrycznej dwóch argumentów i obliczenie odpowiadającego jej wyniku. S-bloki, jako nieliniowe składniki systemu, są kluczowe dla jego bezpieczeństwa (widzieliśmy bowiem w rozdziale 1, jak łatwo można złamać liniowe systemy kryptograficzne, takie jak szyfr Hilla, poprzez atak oparty na znanym tekście jawnym). Jednakże założenia projektowe S-bloków nie są w pełni znane. Wiele osób sugerowało, że S-bloki mogą zawierać ukryte „tylne wyjście”, którymi National Security Agency (NSA, Agencja Bezpieczeństwa Narodowego) mogłyby się przedostawać do systemu i deszyfrować komunikaty, zapewniając jednocześnie o „bezpieczeństwie” systemu DES. Trudno oczywiście obalić takie podejrzenie, jednak jak dotychczas nic nie wskazuje na to, by w rzeczywistości istniały w DES tego typu pułapki.

W 1976 roku National Security Agency zatwierdziła następujące własności S-bloków, które składały się na założenia projektowe:

- P0** Każdy wiersz w dowolnym S-bloku jest permutacją liczb całkowitych  $0, \dots, 15$ .
- P1** Żaden S-blok nie jest liniową ani afinyczną funkcją swoich danych wejściowych.
- P2** Zmiana jednego bitu w danych wejściowych podawanych do S-bloków powoduje zmianę co najmniej dwóch bitów w wyniku.
- P3** Dla dowolnego S-bloku i ciągu wejściowego  $x$ , ciągi  $S(x)$  i  $S(x \oplus 001100)$  różnią się co najmniej dwoma bitami ( $x$  jest tu ciągiem 6-bitowym).

Dwie dodatkowe własności S-bloków zostały przez NSA określone jako „wykonikające z założeń projektowych”:

- P4** Dla dowolnego S-bloku i ciągu wejściowego  $x$  oraz dowolnych  $e, f \in \{0, 1\}$ ,  $S(x) \neq S(x \oplus 11ef00)$ .
- P5** Jeśli w jakimkolwiek S-bloku ustalimy jeden bit wejściowy i popatrzymy na wartość jednego ustalonego bitu wyjściowego, to liczba danych wejściowych, dla których wybrany bit wyjściowy jest równy 0, będzie „bliska” liczbie danych wejściowych, dla których wartość tego bitu jest równa 1. (Zauważmy, że jeśli ustalimy wartość pierwszego albo szóstego bitu wejściowego, to 16 danych wejściowych spowoduje, że dany bit wyjściowy przybierze wartość 0 i 16 danych wejściowych spowoduje, że tą wartością będzie 1. Dla bitów od drugiego do piątego w ciągu wejściowym tak nie będzie, ale otrzymany rozkład będzie „bliski” rozkładowi jednostajnemu. Dokładniej, jeśli w dowolnym S-bloku ustalimy wartość pewnego bitu wejściowego, to liczba danych wejściowych, dla których ustalony bit wyjściowy przyjmie wartość 0 (lub 1), zawiera się zawsze między 13 i 19).

Do publicznej wiadomości nie dotarła żadna informacja o stosowaniu innych kryteriów przy konstrukcji S-bloków.

Najistotniejsze zastrzeżenie kierowane pod adresem systemu DES dotyczy wielkości przestrzeni kluczy ( $2^{56}$  elementów), zbyt małej, by mogła gwarantować pełne bezpieczeństwo. Można było się obawiać powstania różnych wyspecjalizowanych maszyn, przeznaczonych do ataku opartego na znanym tekście jawnym,

które w istocie rzeczy wykonywałyby wyczerpujące przeszukiwanie przestrzeni kluczy. Maszyna taka, wychodząc od 64-bitowego tekstu jawnego  $x$  i odpowiadającego mu tekstu zaszyfrowanego  $y$  badałaby każdy możliwy klucz, aż do natrafienia na taki klucz  $K$ , dla którego spełniony byłby warunek  $e_K(x) = y$  (zauważmy, że takich kluczy może być więcej niż jeden).

Już w 1977 roku Diffie i Hellman wskazywali na możliwość zbudowania układu scalonego VLSI, który mógłby sprawdzić  $10^6$  kluczy na sekundę. Maszyna wyposażona w  $10^6$  takich układów potrzebowałaby jednego dnia, by przetestować całą przestrzeń kluczy. Według ich szacunku urządzenie takie kosztowałyby ok. 20 milionów dolarów.

Podczas jednej z sesji CRYPTO'83 Michael Wiener przedstawił bardzo szczegółowy projekt maszyny poszukującej klucza. Jej zasadniczym elementem jest układ scalony poszukujący klucza, działający w systemie potokowym, co umożliwia jednoczesne wykonywanie 16 szyfrowań. Ten układ jest w stanie testować  $5 \cdot 10^7$  kluczy na sekundę, a przy współczesnej technologii koszt jednego takiego układu wynosiłby ok. 10,5 dolara. Płytkę zawierającą 5760 układów można byłoby zbudować za 100 000 dolarów, a taka płytka umożliwiałaby znalezienie klucza systemu DES średnio w ciągu 1,5 dnia; maszyna wykorzystująca 10 zestawów kosztowałaby 1 000 000 dolarów, ale przy jej użyciu czas poszukiwania klucza zmalałby do ok. 3,5 godziny.

---

### 3.4. DES w praktyce

Chociaż opis systemu DES jest dość długi, jest to system łatwy do implementacji, zarówno na poziomie sprzętowym, jak i programistycznym. Jedyną wykonywaną operacją arytmetyczną jest różnica symetryczna (alternatywa wykluczająca) ciągów bitów. Funkcję rozszerzenia  $E$ , S-bloki, permutacje IP i P oraz obliczanie  $K_1, K_2, \dots, K_{16}$  można realizować w stałym czasie, przeszukując tablice (na poziomie programistycznym) lub budując układ elektroniczny.

Współczesne implementacje sprzętowe umożliwiają niezwykle szybkie działanie. Podczas CRYPTO'92 firma Digital Equipment Corporation ogłosiła wyprowadkowanie układu scalonego złożonego z 50 000 tranzystorów, który może szyfrować z prędkością 1 Gbit/sek, przy taktowaniu zegarem o częstotliwości 250 MHz! Układ kosztuje ok. 300 dolarów. W 1991 roku istniało 45 implementacji sprzętowych lub w postaci oprogramowania firmowego systemu DES, uznanych przez NSA.

Bardzo ważną dziedzinę zastosowań systemu DES stanowią transakcje bankowe, gdzie wykorzystuje się standardy opracowane przez American Bankers Association (Amerykańskie Stowarzyszenie Bankowców). DES jest używany do szyfrowania osobistych numerów identyfikacyjnych (PIN) oraz operacji na kontach wykonywanych za pośrednictwem bankomatów. Także w systemie rozliczeń międzybankowych (Clearing House Interbank Payments System, CHIPS) stosuje się DES do potwierdzania transakcji; łącznie wartość tych transakcji przewyższa

tygodniowo kwotę  $1,5 \cdot 10^{12}$  dolarów. Wreszcie DES jest powszechnie używany w instytucjach rządowych, takich jak Departament Energii, Departament Sprawiedliwości czy Federalny System Rezerw.

### 3.4.1. Tryby działania DES

Opracowano cztery tryby działania DES: *tryb elektronicznej książki kodowej* (ang. *electronic codebook mode*, ECB), *tryb szyfrowego sprzężenia zwrotnego* (ang. *cipher feedback mode*, CFB), *tryb wiązania bloków szyfrowych* (ang. *cipher block chaining mode*, CBC) oraz *tryb wyjściowego sprzężenia zwrotnego* (ang. *output feedback mode*, OFB).

Tryb ECB odpowiada zwykłemu szyfrowi blokowemu: mając dany ciąg  $x_1x_2\dots$  64-bitowych bloków tekstu jawnego, szyfruje się każdy element  $x_i$  za pomocą tego samego klucza  $K$ , otrzymując w wyniku ciąg bloków tekstu zaszyfrowanego  $y_1y_2\dots$

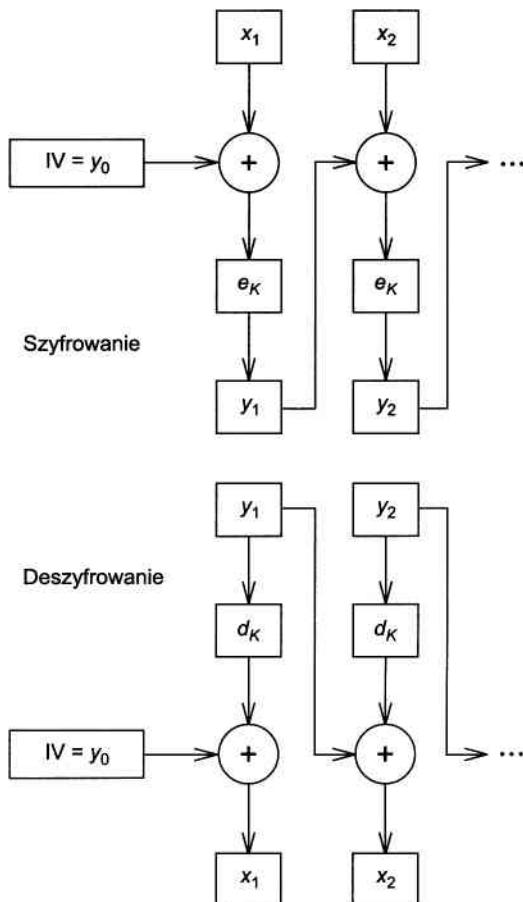
W trybie CBC przed zaszyfrowaniem kolejnego bloku tekstu jawnego oblicza się jego różnicę symetryczną z otrzymanym w poprzednim kroku blokiem tekstu zaszyfrowanego danym kluczem  $K$ . Mówiąc bardziej formalnie, zaczynamy od 64-bitowego *wektora początkowego IV* (ang. *initialization vector*) i definiujemy  $y_0 = \text{IV}$ . Następnie budujemy bloki  $y_1, y_2, \dots$  zgodnie z regułą  $y_i = e_K(y_{i-1} \oplus x_i)$ ,  $i \geq 1$ . Działanie algorytmu DES w trybie CBC ilustruje rysunek 3.4.

W trybach OFB i CFB jest generowany klucz strumieniowy, a następnie jest obliczana różnica symetryczna klucza i tekstu jawnego (w tych trybach algorytm działa zatem jak szyfr strumieniowy, patrz p. 1.1.7). Algorytm DES w trybie OFB jest w istocie synchronicznym szyfrem strumieniowym. Klucz strumieniowy powstaje w wyniku kolejnych szyfrowań 64-bitowego wektora początkowego IV. Definiujemy  $z_0 = \text{IV}$ , po czym obliczamy klucz strumieniowy  $z_1z_2\dots$  z użyciem reguły  $z_i = e_K(z_{i-1})$ ,  $i \geq 1$ . Ciąg tekstu jawnego  $x_1x_2\dots$  jest szyfrowany przez obliczenie  $y_i = x_i \oplus z_i$ ,  $i \geq 1$ .

W trybie CFB zaczynamy od  $y_0 = \text{IV}$  (64-bitowego wektora początkowego) i tworzymy kolejny element  $z_i$  klucza strumieniowego, szyfrując wcześniejszy blok tekstu zaszyfrowanego:  $z_i = e_K(y_{i-1})$ ,  $i \geq 1$ . Podobnie jak w trybie OFB,  $y_i = x_i \oplus z_i$ ,  $i \geq 1$ . Działanie algorytmu DES w trybie CFB ilustruje rysunek 3.5 (zauważmy, że w CFB i OFB funkcję szyfrującą systemu DES, czyli  $e_K$ , stosuje się zarówno do szyfrowania, jak i do deszyfrowania).

Istnieją także modyfikacje trybów OFB i CFB, zwane  $k$ -bitowymi trybami sprzężenia zwrotnego ( $1 \leq k \leq 64$ ). Wyżej opisaliśmy tryby 64-bitowe. W praktyce często używa się trybów 1-bitowych i 8-bitowych, gdy szyfruje się dane bit po bicie lub bajt po bajcie.

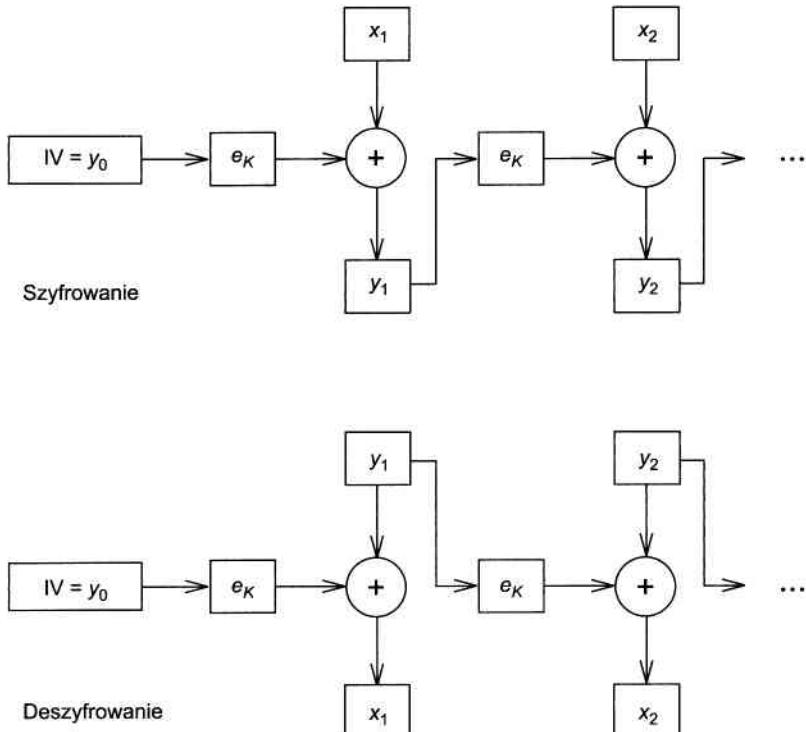
Każdy ze stosowanych trybów ma swoje wady i zalety. Gdy w ECB lub OFB zmieni się jeden 64-bitowy blok  $x_i$  tekstu jawnego, zmiana spowoduje modyfikację odpowiadającego mu bloku tekstu zaszyfrowanego  $y_i$ , lecz nie wpłynie to na inne bloki tekstu zaszyfrowanego. W niektórych sytuacjach taki efekt może się okazać pożądany. Tryb OFB jest często stosowany, na przykład, do szyfrowania transmisji satelitarnych.



RYSUNEK 3.4. Tryb CBC

Natomiast gdy w trybach CBC lub CFB ulegnie zmianie blok  $x_i$  tekstu jawnego, wpłynie to nie tylko na  $y_i$ , ale także na wszystkie następujące po nim bloki tekstu zaszyfrowanego. Ta własność powoduje, że tryby CBC i CFB są przydatne wtedy, gdy chodzi o uwierzytelnienie. Dokładniej, tryby te mogą posłużyć do tworzenia kodu uwierzytelniania komunikatów (ang. *message authentication code*, MAC). Taki kod jest dodawany do ciągu bloków tekstu jawnego w celu przekonania Bolka, że ów ciąg rzeczywiście pochodzi od Alicji i nie został po drodze zmieniony przez Oskara. Tak więc MAC gwarantuje integralność (i autentyczność) wiadomości (choć, rzecz jasna, nie zapewnia poufności).

Opiszmy teraz, jak jest używany tryb CBC do stworzenia kodu uwierzytelniania komunikatu. Zaczynamy od wektora początkowego IV złożonego z samych zer. Dalej, działając w trybie CBC, budujemy bloki tekstu zaszyfrowanego  $y_1 \dots y_n$  za pomocą klucza  $K$ . Wreszcie definiujemy  $y_n$  jako MAC. Następnie Alicja przesyła ciąg bloków tekstu jawnego  $x_1 \dots x_n$  wraz z kodem MAC. Bolek



RYSUNEK 3.5. Tryb CFB

po odebraniu  $x_1 \dots x_n$  potrafi odtworzyć  $y_1 \dots y_n$  za pomocą (tajnego) klucza  $K$  i sprawdzić, czy  $y_n$  pokrywa się z tym, co otrzymał jako MAC.

Zauważmy, że Oskar nie jest w stanie wyprodukować poprawnego kodu MAC, ponieważ nie zna klucza  $K$  używanego przez Alicję i Boleka. Ponadto, jeśli przejmie ciąg bloków tekstu jawnego  $x_1 \dots x_n$  i zmieni jeden lub kilka z nich, to jest bardzo mało prawdopodobne, by mógł zmienić także MAC, tak by Bolek go zaakceptował.

Nierzadko konieczne jest zapewnienie zarówno autentyczności, jak i poufności. Można to uczynić w sposób następujący. Najpierw Alicja używa klucza  $K_1$  do utworzenia kodu MAC dla ciągu  $x_1 \dots x_n$ . Następnie, przyjmując, że MAC jest równy  $x_{n+1}$ , szyfruje ciąg  $x_1 \dots x_{n+1}$  za pomocą innego klucza  $K_2$ . Bolek, po odebraniu ciągu wynikowego  $y_1 \dots y_{n+1}$ , najpierw deszyfruje tekst (za pomocą klucza  $K_2$ ), a następnie za pomocą klucza  $K_1$  sprawdza, czy  $x_{n+1}$  jest kodem MAC ciągu  $x_1 \dots x_n$ .

Ten sam efekt można także uzyskać w inny sposób. Alicja mogłaby użyć klucza  $K_1$  do zaszyfrowania ciągu  $x_1 \dots x_n$ , otrzymując pewien ciąg  $y_1 \dots y_n$ . Dalej użyłaby klucza  $K_2$  do tworzenia kodu MAC dla ciągu  $y_1 \dots y_n$ . Wtedy Bolek użyłby najpierw klucza  $K_2$  do sprawdzenia kodu MAC, a potem klucza  $K_1$  do deszyfrowania ciągu  $y_1 \dots y_n$ .

### 3.5. Wymagania czasowe a wymagania pamięciowe – próba kompromisu

W tym rozdziale opiszemy ciekawe podejście będące rozwiązaniem kompromisowym ze względu na wymagania czasowe i wymagania pamięciowe w przypadku ataku opartego na wybranym tekście jawnym. Przypomnijmy, że w takim ataku Oskar dysponuje tekstem jawnym i odpowiadającym mu tekstem zaszyfrowanym za pomocą (nieznanego) klucza  $K$ . Tak więc Oskar jest w posiadaniu  $x$  oraz  $y$ , gdzie  $y = e_K(x)$ , i chce określić  $K$ .

Szczególną cechą tego rozwiązania kompromisowego jest to, że nie zależy ono w żaden sposób od „struktury” DES. Jedynymi istotnymi dla ataku własnościami systemu DES są długości ciągów: 64 bity w przypadku tekstów jawnych i tekstów zaszyfrowanych oraz 56 bitów dla kluczy.

$$\begin{array}{cccccc} X(1,0) & \xrightarrow{g} & X(1,1) & \xrightarrow{g} & \dots & \xrightarrow{g} & X(1,t) \\ X(2,0) & \xrightarrow{g} & X(2,1) & \xrightarrow{g} & \dots & \xrightarrow{g} & X(2,t) \\ \dots & & & & & & \dots \\ X(m,0) & \xrightarrow{g} & X(m,1) & \xrightarrow{g} & \dots & \xrightarrow{g} & X(m,t) \end{array}$$

RYSUNEK 3.6. Obliczanie  $X(i, j)$

Omawialiśmy już koncepcję wyczerpującego przeszukiwania, polegającą na tym, że będąc w posiadaniu pary tekst jawnny – tekst zaszyfrowany, sprawdzamy wszystkie możliwe klucze w liczbie  $2^{56}$ . Do takiego działania nie jest potrzebna pamięć, natomiast trafienie na poprawny klucz wymaga sprawdzenia średnio  $2^{55}$  kluczy. Z drugiej strony, mając w ręku tekst jawnego, Oskar mógłby wykonać obliczenia wstępne, mianowicie obliczyć  $y_K = e_K(x)$  dla wszystkich  $2^{56}$  kluczy  $K$ , a następnie zbudować tablicę par  $(y_K, K)$ , uporządkowanych według pierwszych współrzędnych. Gdy później Oskar zdobędzie tekst zaszyfrowany  $y$  odpowiadający tekowi jawnemu  $x$ , odnajdzie wartość  $y$  w tablicy, a tym samym natychmiast otrzyma klucz  $K$ . Otóż rzeczywiście ustalenie klucza będzie przebiegać w stałym czasie, lecz tym razem pojawiają się ogromne wymagania dotyczące pamięci oraz czasu obliczeń wstępnych. (Zauważmy, że z punktu widzenia łącznego czasu obliczeń takie podejście nie daje żadnych korzyści, gdy zadanie polega na znalezieniu jednego klucza. Skonstruowanie tablicy wymaga co najmniej tyle samo czasu, co wyczerpujące przeszukiwanie. Korzyść pojawia się dopiero wtedy, gdy w pewnym okresie poszukuje się kilku kluczy naraz, gdyż wtedy można w każdym przypadku użyć tej samej tablicy).

Kompromis między czasem a pamięcią pozwala połączyć krótszy czas obliczeń (w porównaniu z wyczerpującym przeszukiwaniem) z mniejszą pamięcią (w porównaniu z przeglądaniem tablicy). Do opisu algorytmu wystarczą dwa parametry  $m$  i  $t$ , które są dodatnimi liczbami całkowitymi. Algorytm wymaga

także funkcji redukcji  $R$ , skracającej ciąg 64-bitowy do 56 bitów. ( $R$  może na przykład po prostu odrzucać 8 bitów). Niech  $x$  będzie ustalonym ciągiem tekstu jawnego o długości 64. Określmy funkcję  $g(K_0) = R(e_{K_0}(x))$  dla danego ciągu bitów  $K_0$  o długości 56. Jak widać, funkcja  $g$  przekształca ciągi 56-bitowe na ciągi 56-bitowe.

Na etapie wstępny Oskar wybiera  $m$  losowych ciągów bitów o długości 56; nazwijmy je  $X(i, 0)$ ,  $1 \leq i \leq m$ . Oskar oblicza  $X(i, j)$  dla  $1 \leq j \leq t$  zgodnie z zależnością rekurencyjną  $X(i, j) = g(X(i, j - 1))$ ,  $1 \leq i \leq m$ ,  $1 \leq j \leq t$ , jak to widać na rysunku 3.6.

Następnie Oskar buduje tablicę par  $T = (X(i, t), X(i, 0))$  uporządkowanych ze względu na pierwszą współrzędną (a więc przechowywana jest tylko pierwsza i ostatnia kolumna macierzy  $X$ ).

W pewnym momencie Oskar zdobywa tekst zaszyfrowany  $y$ , który odpowiada wybranemu tekstowi jawnemu  $x$  (jak poprzednio). Znów jego zadanie polega na ustaleniu klucza. Ma stwierdzić, czy  $K$  występuje w pierwszych  $t$  kolumnach macierzy  $X$ , ale będzie przeglądał tylko tablicę  $T$ .

Załóżmy, że  $K = X(i, t - j)$  dla pewnego  $j$ , takiego że  $1 \leq j \leq t$  (a więc zakładamy, że  $K$  występuje w pierwszych  $t$  kolumnach macierzy  $X$ ). Widać wówczas, że  $g^j(K) = X(i, t)$ , gdzie  $g^j$  oznacza funkcję otrzymaną przez  $j$ -krotną iterację funkcji  $g$ . Zauważmy teraz, że

$$\begin{aligned} g^j(K) &= g^{j-1}(g(K)) \\ &= g^{j-1}(R(e_K(x))) \\ &= g^{j-1}(R(y)). \end{aligned}$$

Przypuśćmy, że obliczyliśmy  $y_j$  dla  $1 \leq j \leq t$  z następującej zależności rekurencyjnej:

$$y_j = \begin{cases} R(y), & \text{gdy } j = 1, \\ g(y_{j-1}), & \text{gdy } 2 \leq j \leq t. \end{cases}$$

Wynika stąd, że  $y_j = X(i, t)$ , jeśli  $K = X(i, t - j)$ . Jednakże równość  $y_j = X(i, t)$  nie zapewnia równości  $K = X(i, t - j)$ , gdyż funkcja  $R$  nie jest różnowartościowa: jej dziedzina ma  $2^{64}$  elementy, podczas gdy zbiór wartości ma ich  $2^{56}$ , a więc średnio przeciwobraz dowolnego ciągu 56-bitowego ma  $2^8 = 256$  elementów. Tak więc, aby przekonać się, czy  $X(i, t - j)$  jest rzeczywiście kluczem, musimy sprawdzić, czy  $y = e_{X(i, t - j)}(x)$ . Nie zachowaliśmy wartości  $X(i, t - j)$ , ale możemy ją łatwo odtworzyć na podstawie znajomości  $X(i, 0)$ , wykonując  $t - j$  razy iterację funkcji  $g$ .

Działanie Oskara jest opisane algorytmem zamieszczonym na rysunku 3.7.

Analizując prawdopodobieństwo zakończenia algorytmu sukcesem, można wykazać, że jeśli  $mt^2 \approx N = 2^{56}$ , to prawdopodobieństwo tego, że  $K = X(i, t - j)$  dla pewnych  $i, j$ , wynosi około  $0,8mt/N$ . Czynnik 0,8 wynika z faktu, że liczby  $X(i, t)$  nie muszą być różne. Sugeruje się, że należy przyjąć  $m \approx t \approx N^{1/3}$  i zbudować około  $N^{1/3}$  tablic, używając do każdej innej funkcji redukcji  $R$ . Związane

```

(1) oblicz  $y_1 = R(y)$ 
(2) for  $j = 1$  to  $t$  do
(3)   if  $y_j = X(i, t)$  dla pewnego  $i$  then
(4)     oblicz  $X(i, t - j)$  z  $X(i, 0)$  za pomocą  $(t - j)$  iteracji
          funkcji  $g$ 
(5)     if  $y = e_{X(i,t-j)}(x)$  then
          przyjmij  $K = X(i, t - j)$  QUIT
(6)   oblicz  $y_{j+1} = g(y_j)$ 

```

RYSUNEK 3.7. Kompromis między czasem i pamięcią w DES

z tym wymagania dotyczące rozmiaru pamięci to  $112 \cdot N^{2/3}$  bitów (ponieważ musimy przechować  $2 \cdot N^{2/3}$  liczb całkowitych, z których każda zajmuje 56 bitów). Łatwo zauważyc, że czas obliczeń przygotowawczych jest rzędu  $O(N)$ .

Nieco trudniej ocenić czas działania algorytmu. Po pierwsze, można zrealizować krok 3 w (oczekiwany) czasie stałym (za pomocą haszowania) lub (w najgorszym przypadku) w czasie  $O(\log m)$  za pomocą przeszukiwania binarnego. Jeśli warunek opisany w kroku 3 nie jest nigdy spełniony (czyli poszukiwania kończą się porażką), to czas przebiegu algorytmu jest rzędu  $O(N^{2/3})$ . Bardziej szczegółowa analiza wykazuje, że nawet po uwzględnieniu czasu realizacji kroków 4 i 5 oczekiwany czas przebiegu algorytmu wzrasta tylko o stały czynnik.

## 3.6. Kryptoanaliza różnicowa

Biham i Shamir opracowali świetnie dziś znaną metodę ataku na system DES, zwaną „kryptoanalizą różnicową”. Jest to atak z wybranym tekstem jawnym. Choć nie dostarcza ona praktycznego sposobu złamania zwykłego, 16-rundowego szyfru DES, przynosi sukces w przypadku, gdy liczba rund szyfrowania jest mniejsza. Na przykład 8-rundowy szyfr DES można złamać na komputerze osobistym w ciągu kilku minut.

Opiszemy teraz podstawowe pomysły, które znalazły tu zastosowanie. Aby przeprowadzić tego rodzaju atak, możemy zaniedbać permutację początkową IP wraz z permutacją do niej odwrotną (nie mają one znaczenia dla kryptoanalizy). Jak stwierdziliśmy wyżej, rozważamy DES ograniczony do  $n$  rund, dla różnych wartości  $n \leq 16$ . W tym kontekście będziemy traktować  $L_0R_0$  jako tekst jawny, a  $L_nR_n$  jako tekst zaszyfrowany  $n$ -rundowym algorytmem DES. (Zauważmy też, że nie odwracamy ciągu  $L_nR_n$ ).

Kryptoanaliza różnicowa zakłada porównanie różnicy symetrycznej (alternatywy wyłączającej) dwóch tekstów jawnych z różnicą symetryczną odpowiadającą im tekstów zaszyfrowanych. Na ogół będziemy rozpatrywać dwa teksty jawnie  $L_0R_0$  i  $L_0^*R_0^*$  ze znaną wartością różnicy symetrycznej  $L'_0R'_0 = L_0R_0 \oplus L_0^*R_0^*$ .

W naszych rozważaniach będziemy zawsze stosować primy ('') do oznaczania różnicy symetrycznej dwóch ciągów bitów.

### DEFINICJA 3.1

Niech  $S_j$  będzie określonym S-blokiem ( $1 \leq j \leq 8$ ). Rozważmy (uporządkowaną) parę ciągów 6-bitowych, powiedzmy  $(B_j, B_j^*)$ . Wówczas  $B_j \oplus B_j^*$  nazwiemy *wejściową różnicą symetryczną*, a  $S_j(B_j) \oplus S_j(B_j^*)$  *wyjściową różnicą symetryczną* (bloku  $S_j$ ).

Zauważmy, że wejściowa różnica symetryczna jest ciągiem 6-bitowym, podczas gdy wyjściowa różnica symetryczna jest ciągiem 4-bitowym.

### DEFINICJA 3.2

Dla dowolnego  $B'_j \in (\mathbb{Z}_2)^6$ ,  $\Delta(B'_j)$  jest zbiorem wszystkich par uporządkowanych  $(B_j, B_j^*)$ , których wejściową różnicą symetryczną jest  $B'_j$ .

Łatwo zauważyc, że każdy ze zbiorów  $\Delta(B'_j)$  ma  $2^6 = 64$  pary oraz

$$\Delta(B'_j) = \{(B_j, B_j \oplus B'_j) : B_j \in (\mathbb{Z}_2)^6\}.$$

Dla każdej pary należącej do  $\Delta(B'_j)$  możemy obliczyć wyjściową różnicę symetryczną bloku  $S_j$  i uzyskany rozkład przedstawić w formie tablicy. Otrzymujemy 64 wyjściowe różnice symetryczne rozłożone na  $2^4 = 16$  możliwych wartości. Podstawą ataku będzie niejednostajność tych rozkładów.

#### Przykład 3.1

Przypuśćmy, że zajmujemy się pierwszym S-blokiem,  $S_1$ , i wejściową różnicą symetryczną 110100. Wówczas

$$\Delta(110100) = \{(000000, 110100), (000001, 110101), \dots, (111111, 001011)\}.$$

Dla każdej pary uporządkowanej ze zbioru  $\Delta(110100)$  obliczamy wyjściową różnicę symetryczną bloku  $S_1$ . Na przykład,  $S_1(000000) = E_{16} = 1110$  oraz  $S_1(110100) = 9_{16} = 1001$ . Zatem wyjściową różnicę symetryczną pary (000000, 110100) jest 0111.

Po wykonaniu tego działania na każdej z 64 par zbioru  $\Delta(110100)$  otrzymamy następujący rozkład wyjściowych różnic symetrycznych:

0000	0001	0010	0011	0100	0101	0110	0111
0	8	16	6	2	0	0	12
1000	1001	1010	1011	1100	1101	1110	1111
6	0	0	0	0	8	0	6

□

W przykładzie 3.1 występuje tylko osiem z 16 możliwych wyjściowych różnic symetrycznych. W tym szczególnym przypadku rozkład jest wysoce niejednorodny. Na ogół jeśli ustalimy S-blok  $S_j$  oraz wejściową różnicę symetryczną  $B'_j$ , to okaże się, że występuje średnio 75–80% wszystkich możliwych wyjściowych różnic symetrycznych.

Dla wygody wprowadzimy notację odpowiednią do opisu otrzymanych rozkładów, a także sposobu ich powstawania.

### DEFINICJA 3.3

Dla  $1 \leq j \leq 8$  oraz dla 6-bitowych ciągów  $B'_j$  i 4-bitowych  $C'_j$  definiujemy:

$$IN_j(B'_j, C'_j) = \{B_j \in \mathbb{Z}_2^6 : S_j(B_j) \oplus S_j(B_j \oplus B'_j) = C'_j\}$$

oraz

$$N_j(B'_j, C'_j) = |IN_j(B'_j, C'_j)|.$$

$N_j(B'_j, C'_j)$  zlicza pary o wejściowej różnicę symetrycznej równej  $B'_j$ , których wyjściowa różnica symetryczna dla S-bloku  $S_j$  jest równa  $C'_j$ . Same te pary, mające wskazaną wejściową różnicę symetryczną i dające określona wyjściową różnicę symetryczną, można odczytać ze zbioru  $IN_j(B'_j, C'_j)$ . Zbiór ten można rozbić na  $N_j(B'_j, C'_j)/2$  par, z których każda ma (wejściową) różnicę symetryczną równą  $B'_j$ .

Zauważmy, że rozkład stablicowany w przykładzie 3.1 składa się z wartości  $N_1(110100, C'_1)$ ,  $C'_1 \in (\mathbb{Z}_2)^4$ . Na rysunku 3.8 jest zamieszczona lista zbiorów  $IN_1(110100, c'_1)$ .

Dla każdego z ośmiu S-bloków istnieją 64 możliwe wejściowe różnice symetryczne, można zatem obliczyć 512 rozkładów, które można bez trudu stablicować za pomocą komputera.

Przypomnijmy, że dane wejściowe dla S-bloków w  $i$ -tej rundzie pojawiają się w postaci  $B = E \oplus J$ , gdzie  $E = E(R_{i-1})$  jest rozszerzeniem  $R_{i-1}$ , a  $J = K_i$  składa się z bitów klucza w  $i$ -tej rundzie. Wejściową różnicę symetryczną (dla wszystkich ośmiu S-bloków) można teraz obliczyć następująco:

$$B \oplus B^* = (E \oplus J) \oplus (E^* \oplus J) = E \oplus E^*.$$

Warto tu mocno podkreślić, że wejściowa różnica symetryczna nie zależy od bitów klucza (choć wyjściowa różnica symetryczna z pewnością od nich zależy).

Zapiszemy każdy ze składników  $B$ ,  $E$  i  $J$  w postaci konkatenacji ośmiu ciągów 6-bitowych:

$$B = B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8$$

$$E = E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8$$

$$J = J_1 J_2 J_3 J_4 J_5 J_6 J_7 J_8,$$

i podobnie zrobimy z  $B^*$  i  $E^*$ . Założymy przez chwilę, że znamy wartości  $E_j$  i  $E_j^*$  dla pewnego  $j$ ,  $1 \leq j \leq 8$  oraz wartość wyjściowej różnicy symetrycznej dla  $S_j$ ,

Wynik różnicy symetrycznej	Możliwe argumenty wejściowe
0000	
0001	000011,001111,011110,011111 101010,101011,110111,111011
0010	000100,000101,001110,010001 010010,010100,011010,011011 100000,100101,010110,101110 101111,110000,110001,111010
0011	000001,000010,010101,100001 110101,110110
0100	010011,100111
0101	
0110	
0111	000000,001000,001101,010111 011000,011101,100011,101001 101100,110100,111001,111100
1000	001001,001100,011001,101101 111000,111101
1001	
1010	
1011	
1100	
1101	000110,010000,010110,011100 100010,100100,101000,110010
1110	
1111	000111,001010,001011,110011 111110,111111

**RYSUNEK 3.8. Możliwe argumenty wejściowe dla wejściowej różnicy symetrycznej z 110100**

czyli  $C'_j = S_j(B_j) \oplus S_j(B_j^*)$ . Wówczas musi być spełniony warunek

$$E_j \oplus J_j \in IN_j(E'_j, C'_j),$$

gdzie  $E'_j = E_j \oplus E_j^*$ .

Zdefiniujmy zbiór  $test_j$  w sposób następujący:

#### DEFINICJA 3.4

Niech  $E_j$  i  $E_j^*$  będą ciągami 6-bitowymi i niech  $C'_j$  będzie ciągiem 4-bitowym. Definiujemy

$$test_j(E_j, E_j^*, C'_j) = \{B_j \oplus E_j : B_j \in IN_j(E'_j, C'_j)\},$$

gdzie  $E'_j = E_j \oplus E_j^*$ .

Mówiąc pogłównie, bierzymy różnice symetryczne  $E_j$  z każdym elementem zbioru  $IN_j(E'_j, C'_j)$ .

Następujące twierdzenie jest bezpośrednim wnioskiem z powyższych rozważań.

### **TWIERDZENIE 3.1**

Niech  $E_j$  i  $E_j^*$  będą ciągami wejściowymi dla S-bloku  $S_j$  i niech  $C'_j$  będzie wyjściową różnicą symetryczną dla  $S_j$ . Jeśli  $E'_j = E_j \oplus E_j^*$ , to bity klucza  $J_j$  występują w zbiorze  $test_j(E_j, E_j^*, C'_j)$ .

Zauważmy, że w zbiorze  $test_j(E_j, E_j^*, C'_j)$  znajdzie się dokładnie  $N_j(E'_j, C'_j)$  ciągów 6-bitowych; poprawna wartość  $J_j$  musi być jedną z tych możliwości.

#### *Przykład 3.2*

Niech  $E_1 = 000001$ ,  $E_1^* = 110101$  oraz  $C'_1 = 1101$ . Mamy  $N_1(110100, 1101) = 8$ , więc w zbiorze  $test_1(000001, 110101, 1101)$  jest dokładnie 8 ciągów bitów. Z rysunku 3.8 wnioskujemy, że

$$\begin{aligned} IN_1(110100, 1101) &= \\ &= \{000110, 010000, 010110, 011100, 100010, 100100, 101000, 110010\}. \end{aligned}$$

Stąd

$$\begin{aligned} test_1(000001, 110101, 1101) &= \\ &= \{000111, 010001, 010111, 011101, 100011, 100101, 101001, 110011\}. \end{aligned}$$

□

Jeśli mamy do dyspozycji drugą taką trójkę  $E_1, E_1^*, C'_1$ , możemytrzymać drugi zbiór  $test_1$  możliwych wartości bitów klucza w  $J_1$ . Rzeczywista wartość  $J_1$  musi znajdować się w części wspólnej tych dwóch zbiorów. Mając takich trójkę więcej, możemy szybko wyznaczyć wartości bitów klucza w  $J_1$ . Jednym ze sposobów jest utrzymywanie macierzy 64 liczników, które reprezentują 64 możliwości dla sześciu bitów w  $J_1$ . Za każdym razem, gdy odpowiednie bity klucza pojawiają się w zbiorze  $test_1$  dla danej trójki, związany z nimi licznik zwiększa się o jeden. Należy mieć nadzieję, że po przebadaniu  $t$  trójkę znajdziemy dokładnie jeden licznik, zawierający liczbę  $t$ ; będzie on reprezentował prawdziwą wartość bitów klucza w  $J_1$ .

#### **3.6.1. Atak na 3-rundowy szyfr DES**

Popatrzmy teraz, jak pomysły z poprzedniego podrozdziału mogą znaleźć zastosowanie w ataku z wybranym tekstem jawnym na 3-rundowy szyfr DES. Zaczniemy od pary tekstów jawnego i odpowiadających im tekstów zaszyfrowanych:

$L_0R_0, L_0^*R_0^*, L_3R_3$  i  $L_3^*R_3^*$ . Możemy zapisać  $R_3$  w sposób następujący:

$$\begin{aligned} R_3 &= L_2 \oplus f(R_2, K_3) \\ &= R_1 \oplus f(R_2, K_3) \\ &= L_0 \oplus f(R_0, K_1) \oplus f(R_2, K_3). \end{aligned}$$

Podobnie można wyrazić  $R_3^*$ , stąd

$$R'_3 = L'_0 \oplus f(R_0, K_1) \oplus f(R_0^*, K_1) \oplus f(R_2, K_3) \oplus f(R_2^*, K_3).$$

Załóżmy teraz, że wybraliśmy teksty jawne tak, by  $R_0 = R_0^*$ , a więc

$$R'_0 = 00\dots0.$$

Wówczas  $f(R_0, K_1) = f(R_0^*, K_1)$  i w konsekwencji

$$R'_3 = L'_0 \oplus f(R_2, K_3) \oplus f(R_2^*, K_3).$$

Otoż teraz znamy  $R'_3$ , gdyż możemy je obliczyć na podstawie dwóch tekstów zaszyfrowanych, i znamy także  $L'_0$ , gdyż możemy je obliczyć na podstawie dwóch tekstów jawnych. To oznacza, że możemy obliczyć  $f(R_2, K_3) \oplus f(R_2^*, K_3)$  z równania

$$f(R_2, K_3) \oplus f(R_2^*, K_3) = R'_3 \oplus L'_0.$$

Dalej,  $f(R_2, K_3) = P(C)$  oraz  $f(R_2^*, K_3) = P(C^*)$ , gdzie  $C$  i  $C^*$  oznaczają, odpowiednio, dwa ciągi wyjściowe ośmiu S-bloków (przypomnijmy, że  $P$  jest ustaloną i publicznie znaną permutacją). Stąd

$$P(C) \oplus P(C^*) = R'_3 \oplus L'_0$$

i w rezultacie

$$C' = C \oplus C^* = P^{-1}(R'_3 \oplus L'_0). \quad (3.1)$$

To jest wyjściowa różnica symetryczna dla ośmiu S-bloków w trzeciej rundzie.

Znamy także  $R_2 = L_3$  i  $R_2^* = L_3^*$  (stanowią one bowiem część tekstów zaszyfrowanych), możemy więc obliczyć

$$E = E(L_3)$$

(3.2)

oraz

$$E^* = E(L_3^*), \quad (3.3)$$

korzystając z publicznie znanej funkcji rozszerzenia  $E$ . To są dane wejściowe dla S-bloków w trzeciej rundzie. Znamy zatem teraz  $E$ ,  $E^*$  oraz  $C'$  dla trzeciej rundy i podobnie jak w poprzednim podrozdziale możemy zbudować zbiory  $test_1, \dots, test_8$  możliwych wartości bitów klucza w  $J_1, \dots, J_8$ .

Dane wejściowe:  $L_0R_0$ ,  $L_0^*R_0^*$ ,  $L_3R_3$  oraz  $L_3^*R_3^*$ , gdzie  $R_0 = R_0^*$

- (1) oblicz  $C' = P^{-1}(R'_3 \oplus L'_0)$
- (2) oblicz  $E = E(L_3)$  oraz  $E^* = E(L_3^*)$
- (3) **for**  $j = 1$  **to**  $8$  **do**  
oblicz  $test_j(E_j, E_j^*, C'_j)$

### RYSUNEK 3.9. Atak różnicowy na 3-rundowy szyfr DES

Na rysunku 3.9 znajduje się opis w pseudokodzie tego algorytmu. W ataku użyjemy wielu takich trójków  $E, E^*, C'$ . Budujemy osiem macierzy liczników, na podstawie których ustalimy 48 bitów  $K_3$ , klucza dla trzeciej rundy. Wszystkie 56 bitów klucza można będzie następnie uzyskać poprzez wyczerpujące przeszukanie wszystkich  $2^{56} = 2^{28} = 256$  możliwości wypełnienia brakujących 8 bitów.

Zilustrujmy to przykładem.

#### Przykład 3.3

Założymy, że mamy trzy pary tekstów jawnych i zaszyfrowanych; teksty jawnie mają określone różnice symetryczne, szyfrowane za pomocą tego samego klucza. Aby otrzymać krótszą postać tekstu, użyjemy zapisu heksadecymalnego:

tekst jawnny	tekst zaszyfrowany
748502CD38451097	03C70306D8A09F10
3874756438451097	78560A0960E6D4CB
486911026ACDF31	45FA285BE5ADC730
375BD31F6ACDF31	134F7915AC253457
357418DA013FEC86	D8A31B2F28BBC5CF
12549847013FEC86	0F317AC2B23CB944

Z pierwszej pary wyliczamy za pomocą równań (3.2) i (3.3) dane wejściowe S-bloków (dla trzeciej rundy). Oto one:

$$E = 0000000001111100000111010000000110100000001100$$

$$E^* = 10111111000001010101011000000101010000001010010.$$

Wyjściową różnicę symetryczną S-bloku obliczamy z równania (3.1):

$$C' = 10010110010111010101101101100111.$$

Z drugiej pary otrzymujemy następujące dane wejściowe dla S-bloku:

$$E = 1010000010111111110100000101010000001011110110$$

$$E^* = 10001010011010100101111010111110010100010101010,$$

podczas gdy wyjściową różnicę symetryczną S-bloku jest ciąg

$$C' = 10011100100111000001111101010110.$$

Dane wejściowe dla S-bloków otrzymane z trzeciej pary są następujące:

$$E = 111011110001010100000110100011110110100101011111$$

$$E^* = 00000101111010011010001010111110101011000000100,$$

a różnica symetryczna S-bloku:

$$C' = 1101010101110101110110100101011.$$

Dalej umieszczamy w tablicach wartości ósmiu macierzy liczników dla każdej z trzech par. Dla ilustracji prześledźmy tę procedurę na przykładzie macierzy liczników dla  $J_1$  i pierwszej pary. Dla tej pary mamy  $E' = 101111$  oraz  $C' = 1001$ . Ponadto

$$IN_1(101111, 1001) = \{000000, 000111, 101000, 101111\}.$$

Z równości  $E_1 = 000000$  wynika, że

$$J_1 \in test_1(000000, 101111, 1001) = \{000000, 000111, 101000, 101111\}.$$

Tak więc zwiększamy wartości 0, 7, 40 i 47 w macierzy liczników dla  $J_1$ .

Poniżej pokazujemy ostateczne tablice. Jeśli uznamy ciągi 6-bitowe za binarne reprezentacje liczb całkowitych z przedziału od 0 do 63, to 64 wartości odpowiadają zapisom liczb  $0, 1, \dots, 63$ . Oto macierze liczników:

$J_1$															
1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0
0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

$J_2$															
0	0	0	0	1	0	3	0	0	1	0	0	1	0	0	0
0	1	0	0	0	0	2	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	1
0	0	1	1	0	0	0	0	0	1	0	1	0	2	0	0

$J_3$															
0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0
0	0	0	3	0	0	0	0	0	0	0	0	0	0	1	1
0	2	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0

$J_4$															
3	1	0	0	0	0	0	0	0	0	0	2	2	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	1	0	1	1
1	1	1	0	1	0	0	0	0	1	1	1	0	0	1	0
0	0	0	0	1	1	0	0	0	0	0	0	0	2	1	

$J_5$															
0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
0	0	0	0	2	0	0	0	3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0	1	0	0	0	0	2	0

$J_6$															
1	0	0	1	1	0	0	3	0	0	0	0	1	0	0	1
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0
1	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0

$J_7$															
0	0	2	1	0	1	0	3	0	0	0	1	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0	0	2	0	0	0	2	0	0	0	0	1	2	1	1	0
0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0

$J_8$															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1
0	3	0	0	0	0	1	0	0	0	0	0	0	0	0	0

W każdej z ośmiu macierzy liczników jest jeden jedynym licznikiem o wartości 3. Pozycje tych liczników wyznaczają bity klucza w  $J_1, \dots, J_8$ . Są to, odpowiednio, pozycje o numerach: 47, 5, 19, 0, 24, 7, 7, 49. Zapisując te liczby całkowite w systemie binarnym, otrzymujemy  $J_1, \dots, J_8$ :

$$J_1 = 101111$$

$$J_2 = 000101$$

$$J_3 = 010011$$

$$J_4 = 000000$$

$$J_5 = 011000$$

$$J_6 = 000111$$

$$J_7 = 000111$$

$$J_8 = 110001.$$

Patrząc teraz na zestaw podkluczy dla trzeciej rundy, możemy odtworzyć 48 bitów klucza. Klucz  $K$  ma postać

$$\begin{array}{cccc} 0001101 & 0110001 & 01?01?0 & 1?00100 \\ 0101001 & 0000??0 & 111?11? & ?100011 \end{array}$$

Pominęliśmy tu bity parzystości, a znak zapytania „?” zastępuje nieznany bit klucza. Kompletnym kluczem (w systemie heksadecymalnym, łącznie z bitami parzystości) jest ciąg

1A624C89520DEC46.

□

### 3.6.2. Atak na 6-rundowy szyfr DES

Opiszemy teraz, jak rozszerzyć poprzednie rozważania, by uzyskać metodę probabilistycznego ataku na 6-rundowy szyfr DES. Pomyśl polega przede wszystkim na umiejętności dobiorze pary tekstów jawnych z określona różnicą symetryczną, aby następnie wyznaczyć prawdopodobieństwa odpowiedniego ciągu różnic symetrycznych powstających w kolejnych rundach szyfrowania. Musimy teraz zdefiniować pewne ważne pojęcie.

#### DEFINICJA 3.5

Niech  $n \geq 1$  będzie liczbą całkowitą. Mianem  $n$ -rundowej charakterystyki określmy listę postaci

$$L'_0, R'_0, L'_1, R'_1, p_1, \dots, L'_n, R'_n, p_n,$$

spełniającą następujące warunki:

$$(1) L'_i = R'_{i-1} \text{ dla } 1 \leq i \leq n.$$

(2) Niech  $1 \leq i \leq n$  oraz założymy, że  $L_{i-1}, R_{i-1}$  i  $L^*_{i-1}, R^*_{i-1}$  są tak dobrane, że  $L_{i-1} \oplus L^*_{i-1} = L'_{i-1}$  oraz  $R_{i-1} \oplus R^*_{i-1} = R'_{i-1}$ . Przypuśćmy ponadto, że  $L_i, R_i$  oraz  $L^*_i, R^*_i$  są obliczane w jednej rundzie szyfrowania. Wówczas prawdopodobieństwo tego, że  $L_i \oplus L^*_i = L'_i$  oraz  $R_i \oplus R^*_i = R'_i$ , jest równe  $p_i$ . (Zauważmy, że prawdopodobieństwo jest liczone względem wszystkich możliwych 48-wyrazowych ciągów  $J = J_1 \dots J_8$ ).

Prawdopodobieństwem charakterystyki nazywamy iloczyn  $p = p_1 \cdot \dots \cdot p_n$ .

**UWAGA** Przypuśćmy, że wybieramy  $L_0, R_0$  i  $L^*_0, R^*_0$  tak, by  $L_0 \oplus L^*_0 = L'_0$  oraz  $R_0 \oplus R^*_0 = R'_0$ , po czym stosujemy  $n$ -rundowy szyfr DES, otrzymując  $L_1, \dots, L_n$  i  $R_1, \dots, R_n$ . Nie możemy wówczas twierdzić, że prawdopodobieństwo tego, że  $L_i \oplus L^*_i = L'_i$  i  $R_i \oplus R^*_i = R'_i$  dla wszystkich  $i$  ( $1 \leq i \leq n$ ) jest równe  $p_1 \cdot \dots \cdot p_n$ , ponieważ 48-bitowe ciągi w układzie podkluczów  $K_1, \dots, K_n$  nie są wzajemnie niezależne. (Gdyby owe  $n$  ciągów 48-bitowych było wybieranych niezależnie, twierdzenie byłoby prawdziwe). Mimo to możemy się spodziewać, że liczba  $p_1 \cdot \dots \cdot p_n$  będzie stosunkowo dobrze przybliżać rzeczywiste prawdopodobieństwo.

Musimy także zauważyć, że prawdopodobieństwa  $p_i$  występujące w charakterystyce są określone względem dowolnej (choć ustalonej) pary tekstów jawnych,

mających określoną różnicę symetryczną, przy czym 48 bitów klucza w jednej rundzie szyfrowania za pomocą DES przebiega wszystkie  $2^{48}$  możliwości. Lecz kryptoanalytyk próbuje odtworzyć ustalony (choć nieznany) klucz. Będzie on wybierał teksty jawne losowo (wśród tych, które mają odpowiednią różnicę symetryczną) w nadziei, że prawdopodobieństwa zgodności różnic symetrycznych w  $n$  rundach szyfrowania z różnicami symetrycznymi wskazanymi w charakterystyce są bliskie wartościom  $p_1, \dots, p_n$ , odpowiednio.

Na rysunku 3.10 jest pokazany prosty przykład: 1-rundową charakterystykę, która stanowiła podstawę ataku na 3-rundowy **DES** (jak poprzednio używamy reprezentacji heksadecymalnej). Inną charakterystykę 1-rundową widać na rysunku 3.11.

$$\begin{array}{ll} L'_0 = \text{dowolnie} & R'_0 = 00000000_{16} \\ L'_1 = 00000000_{16} & R'_1 = L'_0 \end{array} \quad p = 1$$

### RYSUNEK 3.10. Charakterystyka 1-rundowa

$$\begin{array}{lll} L'_0 = 00000000_{16} & R'_0 = 60000000_{16} \\ L'_1 = 60000000_{16} & R'_1 = 00808200_{16} & p = 14/64 \end{array}$$

RYSUNEK 3.11. Inna charakterystyka 1-rundowa

Popatrzmy uważniej na charakterystykę z rysunku 3.11. Pierwszym krokiem, jaki wykonamy po obliczeniu  $f(R_0, K_1)$  i  $f(R_0^*, K_1)$ , będzie rozszerzenie  $R_0$  i  $R_0^*$ . Różnica symetryczna tych dwóch rozszerzeń jest ciąg

001100...0.

Tak więc wejściową różnicą symetryczną dla  $S_1$  jest 001100, natomiast dla każdego z siedmiu pozostałych S-bloków jest nią ciąg 000000. Dla bloków od  $S_2$  do  $S_8$  ciąg 0000 będzie wyjściową różnicą symetryczną, podczas gdy wyjściową różnicą symetryczną dla  $S_1$  będzie 1110 z prawdopodobieństwem 14/64 (gdzie z obliczeń wynika, że  $N_1(001100, 1110) = 14$ ). Otrzymujemy zatem

$$C' = 111000$$

z prawdopodobieństwem 14/64. Stosując P, mamy

$$P(C) \oplus P(C^*) = 00000000100000001000001000000000,$$

co w systemie heksadecymalnym można zapisać w postaci  $00808200_{16}$ . Obliczając różnicę symetryczną tego ciągu z  $L'_0$ , otrzymujemy wymagane  $R'_1$  z prawdopodobieństwem  $14/64$ . Oczywiście zawsze mamy równość  $L'_1 = R'_0$ .

$L'_0 = 40080000_{16}$	$R'_0 = 04000000_{16}$	
$L'_1 = 04000000_{16}$	$R'_1 = 00000000_{16}$	$p = 1/4$
$L'_2 = 00000000_{16}$	$R'_2 = 04000000_{16}$	$p = 1$
$L'_3 = 04000000_{16}$	$R'_3 = 40080000_{16}$	$p = 1/4$

RYSUNEK 3.12. Charakterystyka 3-rundowa

Atak na 6-rundowy szyfr **DES** jest oparty na 3-rundowej charakterystyce przedstawionej na rysunku 3.12. Rozpoczniemy 6-rundowy atak, mając  $L_0 R_0$ ,  $L_0^* R_0^*$ ,  $L_6 R_6$  i  $L_6^* R_6^*$ , przy czym teksty jawne są wybrane tak, by  $L'_0 = 40080000_{16}$  i  $R'_0 = 04000000_{16}$ . Ciąg  $R_6$  możemy przedstawić w sposób następujący:

$$\begin{aligned} R_6 &= L_5 \oplus f(R_5, K_6) \\ &= R_4 \oplus f(R_5, K_6) \\ &= L_3 \oplus f(R_3, K_4) \oplus f(R_5, K_6). \end{aligned}$$

Podobnie można wyrazić  $R_6^*$  i w rezultacie otrzymujemy

$$R'_6 = L'_3 \oplus f(R_3, K_4) \oplus f(R_3^*, K_4) \oplus f(R_5, K_6) \oplus f(R_5^*, K_6). \quad (3.4)$$

(Zwróćmy uwagę na podobieństwo do ataku 3-rundowego).

Znamy  $R'_6$ . Na podstawie charakterystyki szacujemy, że  $L'_3 = 04000000_{16}$  i  $R'_3 = 40080000_{16}$  z prawdopodobieństwem 1/16. Jeśli tak jest w rzeczywistości, to za pomocą funkcji rozszerzenia można obliczyć wejściową różnicę symetryczną dla S-bloków w czwartej rundzie:

$$00100000000000001010000\ldots0.$$

Wszystkie wejściowe różnice symetryczne dla bloków  $S_2$ ,  $S_5$ ,  $S_6$ ,  $S_7$  oraz  $S_8$  są równe 000000, zatem dla wszystkich tych pięciu bloków wyjściową różnicą symetryczną w czwartej rundzie jest ciąg 0000. Oznacza to, że wyjściowe różnice symetryczne w szóstej rundzie dla tych pięciu bloków możemy obliczyć z równania (3.4). Założymy więc, że obliczamy

$$C'_1 C'_2 C'_3 C'_4 C'_5 C'_6 C'_7 C'_8 = P^{-1}(R'_6 \oplus 04000000_{16}),$$

gdzie każde  $C_i$  jest ciągiem 4-bitowym. Wówczas z prawdopodobieństwem 1/16 ciągi  $C'_2$ ,  $C'_5$ ,  $C'_6$ ,  $C'_7$  i  $C'_8$  będą wyjściowymi różnicami symetrycznymi w szóstej rundzie dla bloków  $S_2$ ,  $S_5$ ,  $S_6$ ,  $S_7$  oraz  $S_8$ . Z obliczeń wynika, że wejściowymi różnicami symetrycznymi dla tych S-bloków w szóstej rundzie będą  $E_2$ ,  $E_5$ ,  $E_6$ ,  $E_7$  i  $E_8$  oraz  $E_2^*$ ,  $E_5^*$ ,  $E_6^*$ ,  $E_7^*$  i  $E_8^*$ , gdzie

$$E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 = E(R_5) = E(L_6)$$

oraz

$$E_1^* E_2^* E_3^* E_4^* E_5^* E_6^* E_7^* E_8^* = E(R_5^*) = E(L_6^*)$$

Dane wejściowe:  $L_0R_0$ ,  $L_0^*R_0^*$ ,  $L_6R_6$  oraz  $L_6^*R_6^*$ , gdzie  
 $L'_0 = 40080000_{16}$  i  $R'_0 = 04000000_{16}$

- (1) oblicz  $C' = P^{-1}(R'_6 \oplus 04000000_{16})$
- (2) oblicz  $E = E(L_6)$  oraz  $E^* = E(L_6^*)$
- (3) **for**  $j \in \{2, 5, 6, 7, 8\}$  **do**  
 oblicz  $test_j(E_j, E_j^*, C'_j)$

RYSUNEK 3.13. Atak różnicowy na 6-rundowy szyfr DES

można obliczyć na podstawie tekstów zaszyfrowanych, jak to widać na rysunku 3.13.

Chcielibyśmy wyznaczyć 30 bitów klucza w  $J_2$ ,  $J_5$ ,  $J_6$ ,  $J_7$  i  $J_8$ , tak jak to zrobiliśmy w ataku 3-rundowym. Kłopot w tym, że prawdopodobieństwo trafienia w wyjściową różnicę symetryczną wynosi jedynie  $1/16$ , co oznacza, że w  $15/16$  wszystkich prób trafimy nie tyle na bity klucza, co na przypadkowe śmieci. Musimy zatem znaleźć sposób na to, by ustalić poprawny klucz na podstawie danych, z których  $15/16$  jest niepoprawnych. Nie wydaje się to zbyt obiecujące, ale okazuje się, że sytuacja nie jest tak beznadziejna, jak na to wygląda.

### DEFINICJA 3.6

Niech  $L_0 \oplus L_0^* = L'_0$  oraz  $R_0 \oplus R_0^* = R'_0$ . Mówimy, że para tekstów jawnych  $L_0R_0$  i  $L_0^*R_0^*$  jest parą *poprawną* względem charakterystyki, jeśli  $L_i \oplus L_i^* = L'_i$  oraz  $R_i \oplus R_i^* = R'_i$  dla wszystkich  $i$ , takich że  $1 \leq i \leq n$ . W przeciwnym przypadku mówimy, że para ta jest *błędna*.

Oczekujemy, że około  $1/16$  wszystkich par będą stanowić pary poprawne, natomiast pozostałe pary będą błędne względem naszej 3-rundowej charakterystyki.

Nasza strategia polega na obliczeniu  $E_j$ ,  $E_j^*$  oraz  $C'_j$  w sposób opisany wyżej, by następnie wyznaczyć  $test_j(E_j, E_j^*, C'_j)$  dla  $j = 2, 5, 6, 7, 8$ . Jeśli zaczniemy od poprawnej pary, to poprawne bity klucza dla każdego  $J_j$  znajdą się w zbiorze  $test_j$ . Jeśli para będzie błędna, uzyskamy niepoprawną wartość  $C'_j$  i należy domniemywać, że każdy ze zbiorów  $test_j$  będzie w istocie zbiorem losowym.

Często można zidentyfikować błędą parę w sposób następujący. Jeśli dla któregokolwiek  $j \in \{2, 5, 6, 7, 8\}$   $|test_j| = 0$ , to z pewnością mamy do czynienia z błędą parą. Otóż mając błędą parę, możemy rozsądnie przyjąć, że prawdopodobieństwo tego, że  $|test_j| = 0$  dla ustalonego  $j$  jest w przybliżeniu równe  $1/5$ . Takie założenie wydaje się uzasadnione tym, że po pierwsze,  $N_j(E'_j, C'_j) = |test_j|$ , a po drugie, jak wspomnieliśmy wcześniej, prawdopodobieństwo tego, że  $N_j(E'_j, C'_j) = 0$  jest w przybliżeniu równe właśnie  $1/5$ . Prawdopodobieństwo tego, że żaden z pięciu zbiorów  $test_j$  nie jest pusty, można oszacować jako  $(0,8)^5 \approx 0,33$ . Zatem prawdopodobieństwo pojawienia się co najmniej jednego zbioru pustego wśród zbiorów  $test_j$  jest równe około  $0,67$ . Dzięki tej prostej obserwacji i wynikającej z niej *operacji filtrowania*, którą przeprowadziliśmy

wyżej, możemy wyeliminować około 2/3 wszystkich błędnych par. Po filtrowaniu stosunek par poprawnych do wszystkich pozostałych par jest w przybliżeniu równy

$$\frac{\frac{1}{16}}{\frac{1}{16} + \frac{15}{16} \cdot \frac{1}{3}} = \frac{1}{6}.$$

### Przykład 3.4

Założmy, że dysponujemy następującą parą tekst jawny–tekst zaszyfrowany:

tekst jawny	tekst zaszyfrowany
86FA1C2B1F51D3BE	1E23ED7F2F553971
C6F21C2B1B51D3BE	296DE2B687AC6340

Wówczas  $L'_0 = 40080000_{16}$  oraz  $R'_0 = 04000000_{16}$ . Obliczając dane wejściowe i wyjściowe dla S-bloków w szóstej rundzie, otrzymujemy:

$j$	$E_j$	$E_j^*$	$C'_j$
2	111100	010010	1101
5	111101	111100	0001
6	011010	000101	0010
7	101111	010110	1100
8	111110	101100	1101

Następnie wyznaczamy zbiory  $test_j$ :

$j$	$test_j$
2	14, 15, 26, 30, 32, 33, 48, 52
5	
6	7, 24, 36, 41, 54, 59
7	
8	34, 35, 48, 49

Widzimy, że zbiory  $test_5$  i  $test_7$  są puste, zatem para jest błędna i zostaje odrzucona w trakcie operacji filtrowania.  $\square$

Przypuśćmy teraz, że mamy parę, dla której  $|test_j| > 0$  dla  $j = 2, 5, 6, 7, 8$ , a więc taką, która przetrwa filtrowanie. (Rzecz jasna, nie umiemy stwierdzić, czy jest to para poprawna czy błędna). Mówimy, że para ta sugeruje 30-bitowy ciąg  $J_2J_5J_6J_7J_8$ , jeśli  $J_j \in test_j$  dla  $j = 2, 5, 6, 7, 8$ . Liczba sugerowanych ciągów jest równa

$$\prod_{j \in \{2, 5, 6, 7, 8\}} |test_j|.$$

Nierzadko liczba sugerowanych ciągów bitów jest bardzo duża (np. większa od 80000).

Założymy, że chcemy stabilicować wszystkie sugerowane ciągi bitów otrzymane z  $N$  par, które nie zostały odrzucone w wyniku operacji filtrowania. Dla każdej poprawnej pary poprawny ciąg  $J_2J_5J_6J_7J_8$  będzie ciągiem sugerowanym; policzymy go mniej więcej  $3N/16$  razy. Ciagi niepoprawne powinny występować dużo rzadziej, ponieważ będą się one pojawiać w zasadzie losowo, a możliwości jest aż  $2^{30}$  (co jest liczbą bardzo dużą).

Zapisanie w tablicy wszystkich sugerowanych ciągów bitów byłoby wielce nieporęczne, użyjemy więc algorytmu wymagającego mniej miejsca i czasu. Każdy zbiór  $test_j$  możemy zakodować w postaci wektora  $T_j$  o długości 64 bitów, w którym  $i$ -ta współrzędna (dla  $0 \leq i \leq 63$ ) jest równa 1, jeśli 6-bitowy ciąg reprezentujący liczbę  $i$  należy do zbioru  $test_j$ ; w przeciwnym przypadku  $i$ -ta współrzędna jest równa 0 (jest to więc w istocie to samo, co reprezentacja w postaci macierzy liczników, której używaliśmy przy 3-rundowym ataku).

Utwórzmy takie wektory dla każdej z pozostałych jeszcze par i nazwijmy je  $T_j^i$ ,  $j = 2, 5, 6, 7, 8$ ,  $1 \leq i \leq N$ . Gdy  $I \subseteq \{1, \dots, N\}$ , mówimy, że  $I$  jest *dopuszczalny*, jeśli dla każdego  $j \in \{2, 5, 6, 7, 8\}$  co najmniej jedna współrzędna wektora

$$\sum_{i \in I} T_j^i$$

jest równa  $|I|$ . Jeśli  $i$ -ta para jest poprawna dla każdego  $i \in I$ , to zbiór  $I$  jest dopuszczalny. Możemy się zatem spodziewać istnienia dopuszczalnego zbioru mającego (w przybliżeniu)  $3N/16$  elementów, który mógłby sugerować tylko poprawne bity klucza. Zbudowanie wszystkich dopuszczalnych zbiorów  $I$  za pomocą algorytmu rekurencyjnego jest już sprawą prostą.

### Przykład 3.5

Aby sprawdzić opisane tu podejście, przeprowadziliśmy próbę komputerową. Wygenerowano losową próbki 120 par tekstów jawnych z określonymi różnicami symetrycznymi, które następnie zaszyfrowano za pomocą tego samego (losowego) klucza. W tablicy 3.1 są przedstawione te pary tekstów zaszyfrowanych oraz odpowiadające im teksty jawne w zapisie heksadecymalnym.

Przy obliczaniu zbiorów dopuszczalnych otrzymujemy  $n_i$  dopuszczalnych zbiorów  $i$ -elementowych, dla następujących wartości  $i$ :

$i$	$n_i$
2	111
3	180
4	231
5	255
6	210
7	120
8	45
9	10
10	1

**TABLICA 3.1. Kryptoanaliza 6-rundowego szyfru DES**

Para	Poprawna?	Tekst jawny	Tekst zaszyfrowany
1	**	86FA1C2B1F51D3BE C6F21C2B1B51D3BE	1E23ED7F2F553971 296DE2B687AC6340
2	**	EDC439EC935E1ACD ADCC39EC975E1ACD	0F847EFE90466588 93E84839F374440B
3	**	9468A0BE00166155 D460A0BE04166155	3D6A906A6566D0BF 3BC3B236398379E1
4	**	D4FF2B18A5AAC8 94F72B18A1AAC8	26B14738C2556BA4 15753FDE86575A8F
5		09D0F2CF277AF54F 49D8F2CF237AF54F	15751F4F11308114 6046A7C863F066AF
6		CBC7157240D415DF 8BCF157244D415DF	7FCDC300FB9698E5 522185DD7E47D43A
7		0D4A1E84890981C1 4D421E848D0981C1	E7C0B01E32557558 912C6341A69DF295
8	**	6CE6B2A9B8194835 2CEEB2A9BC194835	75D52E028A5C48A3 6C88603B48E5A8CE
9	**	799F63C3C9322C1A 399763C3CD322C1A	A6DA322B8F2444B5 6634AA9DF18307F4
10	**	1B36645E381EDF48 5B3E645E3C1EDF48	1F91E295D559091B D094FC12C02C17CA
11		85CA13F50B4ADBB9 C5C213F50F4ADBB9	ED108EE7397DDE0A 3F405F4A3E254714
12	**	7963A8EFD15BC4A1 396BA8EFD55BC4A1	8C71439715A33BA C344C73CC97E4AC4
13		7BCFF7BCA455E65E 3BC7F7BCA055E65E	475A2D0459BCCE62 8E94334AEF359EF8
14		0C505CEDB499218C 4C585CEDB099218C	D3C66239E89CC076 9A316E801EE18EB1
15		6C5EA056CDC91A14 2C56A056C9C91A14	BC7EBA159BCA94E6 67DB935C21FF1A8D
16	**	6622A441A0D32415 262AA441A4D32415	35F8616FEBAB2883 4313E1925F5B64BC
17		C0333C994AFF1C99 803B3C994EFF1C99	D46A4CF1C0221B11 D22B42DB150E2CE8
18		9E7B2974F00E1A6E DE732974F40E1A6E	172D286D9606E6FE 2217A91F8C427D27
19	**	CF592897BFD70C7E 8F512897BBD70C7E	FB892B59E7DCE7EC C328B765E1CC6653
20		E976CF19124A9FA1 A97ECF19164A9FA1	905BF24188509FA6 9ADDBA0C23DD724F
21	**	5C09696E7363675D 1C01696E7763675D	92D60E5C71801A99 DD90908A4FE8168F
22	**	A8145AB3C1B2C7DE E81C5AB3C5B2C7DE	F68FC9F80564847B 51C041B5711B8132
23		47DF6A0BB1787159 07D76A0BB5787159	52E36C4CA22EA5A2 373EAFD503F68DE4
24	*	7CE65464329B4E6D 3CEE5464369B4E6D	832A9D7032015D9F 85E2CE665571E99C

TABLICA 3.1 cd.

Para	Poprawna?	Tekst jawny	Tekst zaszyfrowany
25	**	421FB6AD95791BA7 0217B6AD91791BA7	D1E730BA1DB565E7 188E61735FA4F3CE
26	**	C58E9A361368FFD6 85869A361768FFD6	795EB9D30CAE6879 26D37AC4867ACC61
27	**	DD86B6C74C8EA4E2 9D8EB6C7488EA4E2	CC3B6915C9A348DF 104C2394555645F0
28	**	43DB9D2F483CA585 03D39D2F4C3CA585	E3E4DA503D1B9396 4EA02C0061332443
29	*	855A309F96FEA5EA C552309F92FEA5EA	85AD6E9E352AFafa 929D22370ACAB80D
30	*	AB3CA25B02BD18C8 EB34A25B06BD18C8	0F7D768E9203F786 A1313BC26A99D353
31	**	A9F7A6F4A7C00E06 E9FFA6F4A3C00E06	F26B385E6BA057FD 203D8384F8F54D19
32	**	688B9ACD856D1312 28839ACD816D1312	C41D99C107B4EF76 6CC817CA025A7DAC
33	**	76BF0621C03D4CD9 36B70621C43D4CD9	BBE1F95AFC1E052A 561F4801F2EB0C63
34	**	014CF8D1F981B8EE 4144F8D1FD81B8EE	D27091C4314CBFE8 B7976D6A80E3DB61
35	**	487D66EDE0405F8C 087566EDE4405F8C	8136325C0AEB84CE 8C638BC4495B69A0
36	**	DDCA47093A362521 9DC247093E362521	51040CF16B600FAA 7FC75515AC3CAAF9
37	**	45A9D34A3996F6D9 05A1D34A3D96F6D9	F2004B854AE6C46C 546825016B03D193
38	**	295D2FBFB00875EA 69552FBFB40875EA	A309DF027E69C265 4F633FFB95A0C11E
39		964C8B98D590D524 D6448B98D190D524	1FF1D0271D6F6C18 8CF2D8D401EBFC0F
40		60383D2BAF0836BC 20303D2BAB0836BC	10A82D55FC480640 602346173581EF79
41	**	5CF8D539A22A1CAD 1CF0D539A62A1CAD	92685D806FBE8738 17006DAB2D28081C
42		F95167CAB6565609 B95967CAB2565609	C52E2EB27446054E 0C219F686840E57A
43		49F1C83615874122 09F9C83611874122	2680C8ECDF5E51CD 5022A7B69B4E75EF
44	**	ACB2EC1941B03765 ECBAEC1945B03765	D6B593460098DEC5 D3190A0200FC6B9B
45		CCCC129D5CB55EC0 8CC4129D58B55EC0	3AD22B7EF59E0D5E A48C92CBEC17E430
46	**	917FF8E2EE6B78D5 D177F8E2EA6B78D5	EF847E058DB71724 F243F0554A00E4C5
47	**	51DBCF028E96DE00 11D3CF028A96DE00	574897CA1EE73885 9F0FD0A5B2C2B5FD
48	*	2094942E093463CE 609C942E0D3463CE	59F6A018C6A0D820 799FE001432346C0

TABLICA 3.1 cd.

Para	Poprawna?	Tekst jawny	Tekst zaszyfrowany
49	**	50FB0723D7CD1081 10F30723D3CD1081	16AF758395EA3A7D CDCB23392D144BED
50	*	740815A4F6CDCABB 340015A4F2CDCABB	4A84D2ED4D9351AB 5923D04CE94D6111
51	**	EDA46A1AE93735DC ADAC6A1AED3735DC	0B302A51B7E5476A 5F817F0ABC770E75
52	*	08BC39B766B2C128 48B439B762B2C128	DFB5F3F500BC0100 B7B9FED8AC93EBFA
53	**	A74E29BBA98F2312 E74629BBAD8F2312	A2B352B7F922E8DA D6BC4B89CED2DEAC
54	**	D6F50D31EE4E68AB 96FD0D31EA4E68AB	4D464847065C0938 7554D87AEDCE5634
55	*	06191AA594891CF5 46111AA590891CF5	649C1D084F920F9E BE12A10384365E19
56		5EA7EFD557946962 1EAFEFDD553946962	15E664293F4D77EE E23396A758DC9CE6
57	**	41FB7704781CC88A 01F377047C1CC88A	8ABD385C441FD6CE 06DE8D55777AB65C
58	**	9689B9123F7C5431 D681B9123B7C5431	E1E63120742099BB 1AF88A2CF6649A4A
59		6F25032B4A309BFE 2F2D032B4E309BFE	48FE50DE774288D7 47950691260D5E10
60	**	D8C4B02D8E8BF1E9 98CCB02D8A8BF1E9	F34D565E6AE85683 A4D2DB548622A8E8
61	**	F663E8CCEE86805B B66BE8CCEA86805B	51BD62C9D5D0F0BB D2ABB03CF9D26C0A
62	**	428B29BFDBA838DB 028329BFDBA838DB	006D62A65761089F 9FD73EF6124B0C11
63	**	04BE2D22D81EDC66 44B62D22DC1EDC66	26D99536D99B5707 94144EBDA0CDEB55
64	**	667B779123A3EF80 2673779127A3EF80	5D09CBF2CE7E5A69 5EFF8BPCA7BAA152
65	**	BC86D401D6572438 FC8ED401D2572438	E05572AAA5F6C377 3C670BC455144F61
66	**	6FE5E9547659E401 2FEDE9547259E401	2C465BF6F52F864C B71D106444F95F31
67	**	27D3BAC6453BE3DE 67DBBAC6413BE3DE	8F160E29000461CD 2A6660F46487F885
68	**	1D864E7642A7023A 5D8E4E7646A7023A	65F91EEBFD8A9C05 84761791B3C36661
69	**	5256CA6894707CBA 125ECA6890707CBA	91527F9349ABCF15 30F28F06A7B0A35A
70	**	C05383B8EFCD2BD7 805B83B8EBCD2BD7	710B6EC61BF63E9C 53AC029D8E0179D5
71		50EB21CA13F9A96E 10E321CA17F9A96E	26D95BA4DE4C85CF 8F01A90F638AFFF6
72	**	60EB1229ACD90EDC 20E31229A8D90EDC	3890EE8567782F96 EE404DF7BE537589

TABLICA 3.1 cd.

Para	Poprawna?	Tekst jawnny	Tekst zaszyfrowany
73		8E9A17D17B173B99 CE9217D17F173B99	885C3933627EDEF0 B7ABB6DF5835E962
74		6EC5CD0802C98817 2ECD0806C98817	A985ADFB1FEE013C 0428DE024B7E4604
75	**	1E81712FF1145C06 5E89712FF5145C06	417E667A99B3CFA5 5C24AA056EB1ADBA
76	**	DF3C5C13311AEC7C 9F345C13351AEC7C	BF01675096F1C48A 243D99BCE12DB864
77	**	7C34472994127C2D 3C3C472990127C2D	713915DA311A7CF4 E9733D11D787E20B
78	**	37304DABA75EAFB3 77384DABA35EAFB3	EFB5C37FA0238ADF A728F7407AF958B3
79		D03A16E4C2D8B54B 903216E4C6D8B54B	423FC0AC24CEFEDD 047D8595DB4D372E
80	**	8CED882B5D91832E CCE5882B5991832E	0006E2DE3AF5C2B5 00F6AA9ED614001B
81	**	1BB0E6C79EFBEC41 5BB8E6C79AFBEC41	E9AED4363915775A 655BC48F1FFB5165
82		D41B8346DA9E2252 94138346DE9E2252	34F5E0BCC5B042EA 702D2C48CDBE5173
83	*	02A9D0A0A91F6304 42A1D0A0AD1F6304	E2F1C10E59AF07C5 BDEE6AA00F25F840
84	**	841B3E27C8F0A561 C4133E27CCF0A561	2B288E54D712C92 FF8609C9E7301162
85	**	CDF0A8D6EE909185 8DF8A8D6EA909185	5D661834D1C76324 22034D57D21FFB56
86	**	4C31AC854F44EA34 0C39AC854B44EA34	BD016309AEDB9BB1 C72EEDC4FA1D9312
87		DB3FC0703C972930 9B37C07038972930	296ABCFBF01DF991 CA4700686F9F83A2
88		E4B362BFD6A7CFD1 A4BB62BFD2A7CFD1	20FDAF335F25B1DA 008C24D75E14ACBD
89		F234232A0E0A4A28 B23C232A0A0A4A28	90CFD699F2DEC5BD 2918D3DE0C1B689C
90	**	71265345A5874004 312E5345A1874004	3052CE3CE88710AE 38F0FC685DF30564
91	**	3E6364548C857110 7B6B645488857110	0E8581E42C9FEC6F 4DD1751861EC5529
92	*	464FBEDBD78900A7 0647BEDBD38900A7	90F5F9ADEDED627A 2EF4C540425E339B
93	**	373B75F847480BB0 773375F843480BB0	5408B964F8442D16 805287D52599E9F0
94	**	D714E87810DE97AC 971CE87814DE97AC	4EC4D623108FA909 0AA0725CED10D6A3
95		B9B5932EF54B2C60 F9BD932EF14B2C60	4B438B3CCF36DEC9 054C6A337709280D
96	**	2F283C38D2E4E1DD 6F203C38D6E4E1DD	83515FB6DFEA90B8 09BCC4FF38C78C23

TABLICA 3.1 cd.

Para	Poprawna?	Tekst jawny	Tekst zaszyfrowany
97	**	1EB8ADAA43BBD575 5EB0ADAA47BBD575	21A1E04813616E42 D044BA3F25DFD02A
98	**	3164AA5454D9F991 716CAA5450D9F991	9382C6C1883F1038 5CDFED4FF2117DEC
99		D78C1C5C6F2243D2 97841C5C6B2243D2	1CC EB091E030E6A6 4DA2CD67CC449B21
100		BEE212A7D3CE3D14 FBEA12A7D7CE3D14	2917C207B4D93E0D A01D50E5A2B902D8
101	**	1049117795E98D0FB 504117795A98D0FB	40916A71385C2803 413FD26EF671F46D
102	**	4DDA114D6FEFEEB4 0DD2114D6AFEEEB4	2E2C65E1D5CBAC31 A16FF03BC0913ED6
103		E0BED7B285BF0A77 A0B6D7B281BF0A77	5D9EFEFFF0AD10490 4C6CA1FAC36A8E5B
104	**	0AE1555FA1716214 4AE9555FA5716214	378400BCED39EB81 A1E0C758BD8912C2
105	**	4657C26790FCB354 065FC26794FCB354	588BA079B2E7ED20 DA90827AEED7A41F
106	**	32BD719B0DC1B091 72B5719B09C1B091	F3477C7552BCB05D EFF444449D66BE9E
107	**	0992F8C8C73A9BFE 499AF8C8C33A9BFE	9F3FFD0F158295F6 C138358DCECC8FC7
108		02C3F061A237BBEB 42CBF061A637BBEB	AC28B0307127EA7C 3FF1DAED9E0FCBC5
109	**	80E529E69EDE6827 C0ED29E69ADE6827	1DF1DB7B66BA1AF1 15700151A5804549
110		B55E84630067B8D5 F55684630467B8D5	88321611FF9DA421 90649D7EACF91F9A
111		2749C2EBC603BFF2 6741C2EBC203BFF2	A62B23A7348E2C3A EB760A09C7FF5153
112	**	C4C5E14D4C5D9FF5 84CDE14D485D9FF5	ABC2312FBFD94DF5 D2BB5954E5062D53
113	**	1566BA21F2647E18 556EBA21F6647E18	A247ED988457CB78 5E99F231005F5249
114	**	2D093D426D922F92 6D013D4269922F92	5DF62030B9F23AE9 5D92DA1FA3D07BA1
115		004518468E0C96C3 404D18468A0C96C3	F28D85FF7E84F38F 52541B0443053C57
116	**	437B70A98AE03344 037370A98EE03344	04B3FBF9823B4CF7 14EBEC79DAD3093E
117		2D01F1073D3E375B 6D09F107393E375B	F10B3E1EE356226C 6FF26DA5E3525B62
118	*	66573DD7E0D7F110 265F3DD7E4D7F110	F2F26204C29FE51E 083A4ECE57E429AC
119		0846DB9538155201 484EDB953C155201	F120D0D2AE788057 00CC914A33034782
120		ABB34FC195C820D1 EBBB4FC191C820D1	5F17AE066B50FC81 2858DD63A2FA4B53

Jedynym dopuszczalnym zbiorem 10-elementowym jest zbiór  
 $\{24, 29, 30, 48, 50, 52, 55, 83, 92, 118\}$ .

W rzeczywistości zbiór ten powstał z dziesięciu par poprawnych. Sugeruje on poprawne bity klucza tylko dla  $J_2, J_5, J_6, J_7$  oraz  $J_8$ . Oto one:

$$J_2 = 011001$$

$$J_5 = 110000$$

$$J_6 = 001001$$

$$J_7 = 101010$$

$$J_8 = 100011$$

Zauważmy, że wszystkie zbiory dopuszczalne co najmniej 6-elementowe oraz niemal wszystkie (oprócz trzech) zbiory dopuszczalne 5-elementowe powstają z poprawnych par, gdyż  $\binom{10}{5} = 252$  oraz  $\binom{10}{i} = n_i$  dla  $6 \leq i \leq 10$ .

Tą metodą możemy uzyskać 30 z 56 bitów klucza. Dzięki innej 3-rundowej charakterystyce, przedstawionej na rys. 3.14, można obliczyć następnych 12 bitów, a mianowicie bity z  $J_1$  i  $J_4$ . Brakuje nam zatem jeszcze 14 bitów. Jednakże liczba  $2^{14} = 16384$  jest stosunkowo niewielka, więc w celu ustalenia owszych 14 bitów klucza można wykonać wyczerpujące przeszukanie.

$L'_0 = 00200008_{16}$	$R'_0 = 00000400_{16}$	
$L'_1 = 00000400_{16}$	$R'_1 = 00000000_{16}$	$p = 1/4$
$L'_2 = 00000000_{16}$	$R'_2 = 00000400_{16}$	$p = 1$
$L'_3 = 00000400_{16}$	$R'_3 = 00200008_{16}$	$p = 1/4$

**RYSUNEK 3.14. Inna charakterystyka 3-rundowa**

Kompletny klucz (w zapisie heksagonalnym, łącznie z bitami parzystości) wygląda tak:

34E9F71A20756231.

Jak wspomnieliśmy, tablica 3.1 zawiera wszystkie 120 par. W drugiej kolumnie zaznaczyliśmy jedną gwiazdką \* pary poprawne, a dwiema gwiazdkami \*\* te pary, o których wiemy, że są błędne i które zostały odrzucone przy filtrowaniu. Spośród tych 120 par 73 zostały uznane za błędne w trakcie procesu filtrowania, zatem pozostały z 47 „możliwymi” parami poprawnymi.  $\square$

### 3.6.3. Inne przykłady kryptoanalizy różnicowej

Metodą kryptoanalizy różnicowej można także atakować DES o większej liczbie rund. Szyfr 8-rundowy wymaga  $2^{14}$  wybranych tekstów jawnych, 10-, 12-, 14-

i 16-rundowy DES można złamać, gdy dysponuje się, odpowiednio,  $2^{24}$ ,  $2^{31}$ ,  $2^{39}$  i  $2^{47}$  wybranymi tekstami jawnymi. Gdy liczba rund przekracza 10, atak jest, jak dotąd, praktycznie niewykonalny.

Wiele podstawieniowo-permutacyjnych szyfrów produktowych jest także (w różnym) stopniu podatnych na kryptoanalizę różnicową. Są wśród nich systemy kryptograficzne powstałe w ostatnich latach, jak na przykład FEAL, REDOC-II i LOKI.

---

### 3.7. Uwagi i bibliografia

Smid i Branstad [SB92] napisali dobry artykuł o historii systemu DES. Publikacje Federalnych Standardów Przetwarzania Informacji (*Federal Information Processing Standards*, FIPS) poświęcone DES obejmują: opis DES [NBS77], implementację i użytkowanie DES [NBS81], tryby działania DES [NBS80] oraz uwierzytelnianie komunikatów za pomocą DES [NBS85].

Niektóre własności S-bloków badali Brickell, Moore i Purtill w [BMP87].

W [EB93] można znaleźć opis układu scalonego DEC dla szyfru DES. Maszyna Wienera do przeszukiwania przestrzeni kluczów została przedstawiona w ramach CRYPTO'93 [Wi94].

Rozważania na temat kompromisu między czasem a pamięcią w DES pochodzą z pracy Hellmana [HE80]. Ogólniejszy wariant takiego kompromisu przedstawili Fiat i Naor w [FN91].

Metodę kryptoanalizy różnicowej zawdzięczamy Bihamowi i Shamirowi [BS91] (patrz także [BS93A] oraz ich książka [BS93], gdzie została omówiona również kryptoanaliza innych systemów). Przedstawione tu podejście do kryptoanalizy różnicowej opiera się w dużej mierze na [BS93].

Inną metodę, kryptoanalizę liniową, która może posłużyć do ataku na DES i podobne systemy kryptograficzne, zaproponował Matsui [MA94, MA94A].

O innych systemach podstawieniowo-permutacyjnych można przeczytać w następujących źródłach: LUCIFER [FE73]; FEAL [Mi91]; REDOC-II [CW91] oraz LOKI [BKPS90].

---

### Ćwiczenia

- 3.1. Wykaż, że do deszyfrowania w systemie DES można użyć algorytmu szyfrującego DES, stosując go do tekstu zaszyfrowanego z odwróconym układem podkluczów.
- 3.2. Niech  $\text{DES}(x, K)$  oznacza wynik szyfrowania w systemie DES tekstu jawnego  $x$  z użyciem klucza  $K$ . Przyjmijmy, że  $y = \text{DES}(x, K)$  oraz  $y' = \text{DES}((c(x), c(K)), g)$ , gdzie  $c(\cdot)$  oznacza dopełnienie argumentu<sup>†</sup>. Wykaż, że  $y' = c(y)$ . (Inaczej mó-

<sup>†</sup>Czyli ciąg, który powstaje z argumentu przez zamianę 0 na 1 i 1 na 0 (przyp. tłum.).

wiąc, dopełniając tekst jawnego i klucz, uzyskamy dopełnienie tekstu zaszyfrowanego). Zwróć uwagę, że do dowodu wystarczy „wysokiego poziomu” opis systemu DES; rzeczywista struktura S-bloków i innych składników systemu nie ma tu znaczenia.

- 3.3. Jednym ze sposobów wzmacniania DES jest *dwukrotne szyfrowanie*: mając dwa klucze  $K_1$  i  $K_2$ , definiujemy  $y = e_{K_2}(e_{K_1}(x))$  (mamy więc po prostu produkt DES ze sobą). Jeśli tak się składa, że funkcja szyfrowania  $e_{K_2}$  jest tą samą funkcją co funkcja deszyfrowania  $d_{K_1}$ , to mówimy, że klucze  $K_1$  i  $K_2$  są *dualne*. (Jest to sytuacja wysoce niepożądana przy dwukrotnym szyfrowaniu, gdyż w efekcie końcowym tekst zaszyfrowany jest tożsamy z tekstem jawnym). Klucz jest **samodualny**, jeśli jest swoim kluczem dualnym.

- (a) Wykaż, że jeśli  $C_0$  składa się albo z samych zer, albo z samych jedynek oraz  $D_0$  także składa się albo z samych zer, albo z samych jedynek, to klucz  $K$  jest samodualny.
- (b) Wykaż, że następujące klucze (podane w zapisie heksadecymalnym) są samodualne:

```
0101010101010101  
FEFEFEFEFEFEFEFE  
1F1F1F1FOEOEOE0E  
EOEOEOEOF1F1F1F1
```

- (c) Wykaż, że jeśli  $C_0 = 0101\dots01$  lub  $1010\dots10$  (w zapisie binarnym), to różnicą symetryczną ciągów  $C_i$  i  $C_{17-i}$  jest ciąg  $1111\dots11$ , dla  $1 \leq i \leq 16$  (podobną własność mają ciągi  $D_i$ ).
- (d) Wykaż, że następujące pary (w zapisie heksagonalnym) składają się z kluczy dualnych:

```
E001E001F101F101    01E001E001F101F1  
FE1FFE1FFEOEFEOE    1FFE1FFE0EFEOEFE  
E01FE01FF10EF10E    1FE01FE00EF10EF1
```

- 3.4. Kod uwierzytelniania komunikatów (MAC) można wygenerować zarówno w trybie CBC, jak i trybie CFB. Przypuśćmy, że dla danego ciągu bloków tekstu jawnego  $x_1 \dots x_n$  definiujemy wektor początkowy IV równy  $x_1$ . Następnie szyfrujemy  $x_2 \dots x_n$  za pomocą klucza  $K$  w trybie CFB, otrzymując  $y_1 \dots y_{n-1}$  (zauważmy, że bloków tekstu zaszyfrowanego jest już tylko  $n-1$ ). Wreszcie określamy MAC w postaci  $e_K(y_{n-1})$ . Wykaż, że tak otrzymany MAC jest tożsamy z tym, który utworzyliśmy w p. 3.4.1 w trybie CBC.
- 3.5. Założmy, że użyto szyfru DES do zaszyfrowania ciągu bloków tekstu jawnego  $x_1 \dots x_n$ , otrzymując bloki tekstu zaszyfrowanego  $y_1 \dots y_n$ . Przypuśćmy, że jeden z bloków tekstu zaszyfrowanego, powiedzmy  $y_i$ , przekazano niepoprawnie (a więc na niektórych pozycjach zamieniono 1 na 0 lub na odwrót). Wykaż, że liczba źle odczytyanych bloków tekstu jawnego jest równa 1, gdy szyfrowanie odbyło się w trybie ECB lub OFB, i jest równa 2, jeśli szyfrowano w trybie CBC lub CFB.
- 3.6. W tym ćwiczeniu chodzi o zbadanie uproszczonego kompromisu między czasem i pamięcią dla ataku opartego na wybranym tekście jawnym. Założmy, że mamy

system kryptograficzny, w którym  $\mathcal{P} = \mathcal{C} = \mathcal{K}$  i który zapewnia tajność doskonałą. Musi zatem być tak, że równość  $e_K(x) = e_{K_1}(x)$  pociąga za sobą równość  $K = K_1$ . Niech  $\mathcal{P} = Y = \{y_1, \dots, y_N\}$  i niech  $x$  będzie ustalonym tekstem jawnym. Określmy funkcję  $g : Y \rightarrow Y$  tak, że  $g(y) = e_y(x)$ . Zbudujmy na zbiorze wierzchołków  $Y$  graf skierowany  $G$ , w którym zbiór krawędzi składa się ze wszystkich skierowanych krawędzi postaci  $(y_i, g(y_i))$  dla  $1 \leq i \leq N$ .

- (a) Udowodnij, że  $G$  jest sumą rozłącznych cykli skierowanych.
- (b) Niech  $T$  będzie parametrem określającym pożądany czas. Przypuśćmy, że mamy zbiór  $Z = \{z_1, \dots, z_m\} \subseteq Y$ , taki że dla każdego elementu  $y_i \in Y$  albo  $y_i$  należy do cyklu o długości co najwyżej  $T$ , albo istnieje element  $z_j \neq y_i$ , taki że odległość między  $y_i$  i  $z_j$  w grafie  $G$  nie przekracza  $T$ . Wykaż, że istnieje zbiór  $Z$ , taki że

$$|Z| \leq \frac{2N}{T},$$

a więc  $|Z|$  jest postaci  $O(N/T)$ .

- (c) Dla każdego  $z_j \in Z$  określmy  $g^{-T}(z_j)$  jako ten element  $y_i$ , dla którego  $g^T(y_i) = z_j$ , gdzie  $g^T$  jest  $T$ -krotną iteracją funkcji  $g$ . Zbudujmy tablicę  $X$  złożoną z par uporządkowanych  $(z_j, g^{-T}(z_j))$ , uporządkowaną ze względu na pierwsze współrzędne.

Rysunek 3.15 przedstawia opis w pseudokodzie algorytmu znajdowania klucza  $K$ , dla którego  $y = e_K(x)$ . Udowodnij, że algorytm ten znajduje  $K$  w co najwyżej  $T$  krokach. (Stąd kompromis między czasem i pamięcią jest rzędu  $O(N)$ ).

- (d) Opisz z użyciem pseudokodu algorytm budowania wymaganego zbioru  $Z$  w czasie  $O(NT)$  bez korzystania z macierzy rzędu  $N$ .

```

(1)  $y_{pocz} = y$ 
(2)  $backup = \text{false}$ 
(3) while  $g(y) \neq y_{pocz}$  do
(4)     if  $y = z_j$  dla pewnego  $j$  and not  $backup$  do
(5)          $y = g^{-T}(z_j)$ 
(6)          $backup = \text{true}$ 
        else
(7)          $y = g(y)$ 
(8)      $K = y$ 

```

**RYSUNEK 3.15. Kompromis między czasem i pamięcią**

3.7. Oblicz prawdopodobieństwa następującej charakterystyki 3-rundowej:

$L'_0 = 00200008_{16}$	$R'_0 = 00000400_{16}$	
$L'_1 = 00000400_{16}$	$R'_1 = 00000000_{16}$	$p = ?$
$L'_2 = 00000000_{16}$	$R'_2 = 00000400_{16}$	$p = ?$
$L'_3 = 00000400_{16}$	$R'_3 = 00200008_{16}$	$p = ?$

Dane wejściowe:  $L_0R_0, L_0^*R_0^*, L_4R_4$  oraz  $L_4^*R_4^*$ , gdzie  
 $L'_0 = 20000000_{16}$  i  $R'_0 = 00000000_{16}$

- (1) oblicz  $C' = P^{-1}(R'_4)$
- (2) oblicz  $E = E(L_4)$  oraz  $E^* = E(L_4^*)$
- (3) **for**  $j = 2$  **to**  $8$  **do**  
 oblicz  $test_j(E_j, E_j^*, C'_j)$

### RYSUNEK 3.6. Atak różnicowy na 4-rundowy szyfr DES

- 3.8. Rozważmy różnicowy atak na 4-rundowy **DES** z użyciem następującej charakterystyki, będącej szczególnym przypadkiem charakterystyki przedstawionej na rysunku 3.10:

$L'_0 = 20000000_{16}$	$R'_0 = 00000000_{16}$	
$L'_1 = 00000000_{16}$	$R'_1 = 20000000_{16}$	

$p = 1$

- (a) Założmy, że do obliczenia zbiorów  $test_2, \dots, test_8$  użyto algorytmu przedstawionego na rysunku 3.16. Wykaż, że  $J_j \in test_j$  dla  $2 \leq j \leq 8$ .
- (b) Mając dane pary tekst jawny–tekst zaszyfrowany, znajdź bity klucza  $J_2, \dots, J_8$ .

tekst jawny	tekst zaszyfrowany
18493AC485B8D9A0	E332151312A18B4F
38493AC485B8D9A0	87391C27E5282161
482765DDD7009123	B5DDD8339D82D1D1
682765DDD7009123	81F4B92BD94B6FD8
ABCD098733731FF1	93A4B42F62EA59E4
8BCD098733731FF1	ABA494072BF411E5
13578642AFFEDCB	FDEB526275FB9D94
33578642AFFEDCB	CC8F72AAE685FDB1

- (c) Oblicz cały klucz (pozostaje do ustalenia 14 bitów, które można znaleźć metodą wyczerpującego przeszukiwania).

# 4

---

## System RSA i faktoryzacja

---

### 4.1. Wprowadzenie do kryptografii z kluczem publicznym

W rozważanym do tej pory klasycznym modelu kryptografii Alicja i Bolek potajemnie wybierali klucz  $K$ , z którego wywodziły się reguły szyfrowania  $e_K$  i deszyfrowania  $d_K$ . W systemach kryptograficznych, które dotąd rozważaliśmy,  $d_K$  jest albo taka sama jak reguła  $e_K$ , albo da się łatwo z niej wyprowadzić (np. deszyfrowanie w szyfrze **DES** jest identyczne z szyfrowaniem z odwróconym zestawem podkluczów). Systemy tego typu są znane jako systemy z *kluczem prywatnym*; cechują się tym, że ujawnienie  $e_K$  wystawia system na niebezpieczeństwo.

Jedną z wad systemu z kluczem prywatnym jest to, że zanim nastąpi przesłanie pierwszego tekstu zaszyfrowanego, Alicja i Bolek muszą przekazać sobie klucz, używając do tego bezpiecznego kanału. W praktyce może się to okazać bardzo trudne do osiągnięcia. Jeśli, na przykład, Alicja i Bolek mieszkają daleko od siebie i postanawiają komunikować się za pomocą poczty elektronicznej, to mogą nie znaleźć wystarczająco bezpiecznego kanału komunikacyjnego.

Istotą systemów z *kluczem publicznym* jest możliwość znalezienia takiego systemu kryptograficznego, w którym ustalenie  $d_K$  dla znanej reguły  $e_K$  jest obliczeniowo niewykonalne. Jeśli tak jest, to regułę szyfrowania można ujawnić, choćby publikując ją w książce telefonicznej (stąd nazwa systemu z kluczem publicznym). Zaletą takiego systemu jest to, że Alicja (lub ktokolwiek inny) może wysłać zaszyfrowany komunikat do Bolka (nie podając mu wcześniej klucza tajnego), używając publicznej reguły szyfrowania  $e_K$ , a Bolek będzie jedyną osobą, która potrafi odczytać tekst zaszyfrowany za pomocą swojej tajnej reguły deszyfrowania  $d_K$ .

Działanie systemu ilustruje następująca analogia. Wyobraźmy sobie, że Alicja zamkuje jakiś przedmiot w metalowym pudełku, zamykając je na kłódkę z szyfrem pozostawioną przez Bolka. Wtedy Bolek będzie jedyną osobą, która może ten przedmiot wydostać z pudełka, ponieważ tylko on zna szyfr kłódki.

Pomysł systemu z kluczem publicznym pochodzi od Diffiego i Hellmana, którzy przedstawili go w 1976 roku. Pierwszymi realizatorami takiego systemu byli w 1977 roku Rivest, Shamir i Adleman. Opracowali oni powszechnie znany

**system kryptograficzny RSA**, który przedstawimy w tym rozdziale. Od tej pory powstało wiele systemów z kluczem publicznym wywodzących swoje bezpieczeństwo z różnych problemów obliczeniowych. Najważniejsze z nich to:

**RSA.** Bezpieczeństwo systemu **RSA** wiąże się z trudnością rozkładu dużych liczb na czynniki pierwsze. Opisujemy ten system w podrozdziale 4.3.

**System plecakowy Merkle'a–Hellmana.** Ten i podobne mu systemy opierają swoje bezpieczeństwo na trudności problemu pakowania (NP-zupełnego<sup>1</sup>). Niestety, wykazano, że systemy plecakowe nie gwarantują pełnego bezpieczeństwa (z wyjątkiem systemu kryptograficznego Chor–Rivesta, którym zajmiemy się dalej). System Merkle'a–Hellmana omawiamy w rozdziale 5.

**System McEliece'a.** System kryptograficzny McEliece'a czerpie z algebraicznej teorii kodowania i jest wciąż uważany za system bezpieczny. U jego podstaw leży problem dekodowania kodu liniowego (także NP-zupełny). Patrz rozdział 5.

**System ElGamala.** Bezpieczeństwo systemu kryptograficznego ElGamala wynika z trudności obliczania logarytmów dyskretnych w ciele skończonym. (Patrz rozdział 5).

**System Chora–Rivesta.** To także system typu plecakowego, nadal jednak uznawany za system bezpieczny.

**Systemy kryptograficzne oparte na krzywych eliptycznych.** Systemy te są modyfikacjami innych systemów (jak np. system ElGamala), wymagającymi obliczeń nie w skończonych ciałach, lecz na krzywych eliptycznych. Wydaje się, że systemy te zachowują bezpieczeństwo przy krótszych kluczach, niż wymagają tego inne systemy z kluczem publicznym. (Patrz rozdział 5).

Odnosimy od razu bardzo istotną cechę systemów z kluczem publicznym: nigdy nie mogą one zapewnić bezwarunkowego bezpieczeństwa. Wynika to z tego, że nieprzyjaciel, mając do dyspozycji tekst zaszyfrowany  $y$  i publiczną regułę szyfrowania  $e_K$ , może szyfrować po kolej wszystkie możliwe teksty jawne, dopóki nie trafi na jedyny tekst  $x$ , dla którego  $y = e_K(x)$ . Wówczas tekst  $x$  jest wynikiem deszyfrowania tekstu  $y$ . Z tego powodu będziemy odtąd badać jedynie obliczeniowe bezpieczeństwo systemów z kluczem publicznym.

Przy omawianiu systemów z kluczem publicznym wygodnie jest odwołać się do abstrakcyjnego pojęcia jednokierunkowej funkcji z bocznym wejściem (ang. *trapdoor one-way function*). Zdefiniujmy to pojęcie nieformalnie.

Funkcja szyfrująca Bolka, czyli funkcja  $e_K$ , powinna być łatwa do obliczenia. Stwierdziliśmy już, że funkcja odwrotna (czyli deszyfrująca) powinna być obliczeniowo trudna (dla wszystkich prócz Bolka). Gdy funkcja ma taką własność,

---

<sup>1</sup> Problemy NP-zupełne stanowią obszerną klasę problemów, dla których nie znamy żadnego algorytmu działającego w czasie wielomianowym.

a więc sama jest łatwa do obliczenia, a jednocześnie funkcja do niej odwrotna jest obliczeniowo trudna, mówimy, że jest to funkcja *jednokierunkowa*. Chcemy zatem, by funkcja  $e_K$  była (różnowartościową) funkcją jednokierunkową.

Funkcje jednokierunkowe odgrywają w kryptografii zasadniczą rolę, szczególnie przy konstrukcji systemów kryptograficznych z kluczem publicznym, ale także w innych sytuacjach. Niestety, mimo istnienia wielu funkcji, o których sądzi się, że są jednokierunkowe, nie udało się do dziś udowodnić o żadnej z nich, że istotnie ma tę własność.

Oto przykład funkcji, o której sądzi się, że jest jednokierunkowa. Niech  $n$  będzie iloczynem dwóch dużych liczb pierwszych  $p$  oraz  $q$  i niech  $b$  będzie dodatnią liczbą całkowitą. Określmy funkcję  $f: \mathbb{Z}_n \rightarrow \mathbb{Z}_n$  w sposób następujący:

$$f(x) = x^b \bmod n.$$

(Przy odpowiednim doborze wartości  $n$  i  $b$  otrzymamy funkcję szyfrującą systemu RSA, którą omówimy dokładniej nieco później).

Do zbudowania systemu z kluczem publicznym nie wystarcza funkcja jednokierunkowa. Z punktu widzenia Bolka jednokierunkowa funkcja szyfrowania jest niepożądana, ponieważ chciałby on przecież sprawnie odczytywać nadchodzące komunikaty. Tak więc Bolek musi dysponować *bocznym wejściem*, umożliwiającym łatwe odwrócenie funkcji  $e_K$ . Inaczej mówiąc, Bolek powinien mieć możliwość łatwego deszyfrowania dzięki dodatkowej, tajnej wiedzy o  $K$ . *Jednokierunkową funkcję z bocznym wejściem* nazwiemy taką funkcję jednokierunkową, którą można łatwo odwrócić dzięki znajomości odpowiedniego bocznego wejścia.

W podrozdziale 4.3 dowiemy się, jak znaleźć takie boczne wejście dla wyżej określonej funkcji  $f$ . Doprowadzi nas to do systemu kryptograficznego RSA.

## 4.2. Jeszcze o teorii liczb

Zanim przystąpimy do omówienia działania systemu RSA, musimy przypomnieć parę faktów z arytmetyki modularnej i teorii liczb. Potrzebujemy dwóch podstawowych wyników: algorytmu Euklidesa oraz chińskiego twierdzenia o resztach.

### 4.2.1. Algorytm Euklidesa

Stwierdziliśmy już w rozdziale 1, że  $\mathbb{Z}_n$  jest pierścieniem dla dowolnej liczby całkowitej dodatniej  $n$ . Wykazaliśmy także, iż element  $b \in \mathbb{Z}_n$  ma odwrotność wtedy i tylko wtedy, gdy  $\text{NWD}(b, n) = 1$ , i umówiliśmy się, że  $\phi(n)$  oznacza liczbę liczb całkowitych dodatnich mniejszych od  $n$ , które są względnie pierwsze z  $n$ .

Zbiór reszt modulo  $n$  względnie pierwszych z  $n$  oznaczamy symbolem  $\mathbb{Z}_n^*$ . Nietrudno sprawdzić, że  $\mathbb{Z}_n^*$  jest grupą abelową ze względu na mnożenie. Wiemy też, że mnożenie modulo  $n$  jest łączne i przemienne oraz że 1 jest elementem neutralnym mnożenia. Dla każdego elementu zbioru  $\mathbb{Z}_n^*$  istnieje element odwrotny

względem mnożenia (także należący do  $Z_n^*$ ). Wreszcie,  $Z_n^*$  jest zamknięty ze względu na mnożenie, ponieważ liczba  $xy$  jest względnie pierwsza z  $n$ , gdy  $x$  i  $y$  są względnie pierwsze z  $n$  (spróbuj to wykazać!).

Wiemy zatem, że każdy element  $b \in Z_n^*$  ma element odwrotny  $b^{-1}$ , nie mamy jednak jeszcze efektywnego algorytmu znajdowania go. Taki algorytm istnieje i jest nim tzw. rozszerzony algorytm Euklidesa.

Opiszmy przede wszystkim algorytm Euklidesa w jego zasadniczej postaci, która służy do obliczania największego wspólnego dzielnika dwóch liczb całkowitych dodatnich  $r_0$  i  $r_1$ , gdzie  $r_0 > r_1$ . Algorytm polega na wykonaniu następującego ciągu dzieleniń:

$$\begin{aligned} r_0 &= q_1 r_1 + r_2, & 0 < r_2 < r_1 \\ r_1 &= q_2 r_2 + r_3, & 0 < r_3 < r_2 \\ &\vdots \\ r_{m-2} &= q_{m-1} r_{m-1} + r_m, & 0 < r_m < r_{m-1} \\ r_{m-1} &= q_m r_m. \end{aligned}$$

Nietrudno wykazać, że

$$\text{NWD}(r_0, r_1) = \text{NWD}(r_1, r_2) = \dots = \text{NWD}(r_{m-1}, r_m) = r_m,$$

skąd wynika, że  $\text{NWD}(r_0, r_1) = r_m$ .

Algorytm Euklidesa oblicza największy wspólny dzielnik dwóch liczb, można go więc użyć do ustalenia, czy dodatnia liczba całkowita  $b < n$  ma element odwrotny względem mnożenia modulo  $n$ , zaczynając obliczenia od liczb  $r_0 = n$  i  $r_1 = b$ . Niestety, algorytm nie podaje w tym przypadku wartości elementu odwrotnego (jeśli istnieje).

Załóżmy teraz, że zdefiniowaliśmy ciąg liczb  $t_0, t_1, \dots, t_m$  zgodnie z następującym wzorem rekurencyjnym (w którym liczby  $q_j$  są określone jak wyżej):

$$\begin{aligned} t_0 &= 0 \\ t_1 &= 1 \\ t_j &= t_{j-2} - q_{j-1} t_{j-1} \bmod r_0, \quad \text{gdy } j \geq 2. \end{aligned}$$

Mamy wówczas następujące pozyteczne twierdzenie.

### TWIERDZENIE 4.1

Niech  $q_j$  i  $r_j$  będą zdefiniowane tak jak w algorytmie Euklidesa, natomiast liczby  $t_j$  tak jak w powyższym ciągu rekurencyjnym. Wówczas  $r_j \equiv t_j r_1 \pmod{r_0}$  dla każdego  $j$ , takiego że  $0 \leq j \leq m$ .

**DOWÓD** Przeprowadzimy dowód indukcyjny ze względu na  $j$ . Twierdzenie jest w oczywisty sposób prawdziwe dla  $j = 0$  i  $j = 1$ . Założymy, że jest prawdziwe dla  $j = i - 1$  i  $j = i - 2$ , dla pewnego  $i \geq 2$ ; wykażemy jego prawdziwość dla  $j = i$ . Z założenia indukcyjnego mamy

$$r_{i-2} \equiv t_{i-2} r_1 \pmod{r_0}$$

oraz

$$r_{i-1} \equiv t_{i-1}r_1 \pmod{r_0}.$$

Obliczamy:

$$\begin{aligned} r_i &= r_{i-2} - q_{i-1}r_{i-1} = \\ &\equiv t_{i-2}r_1 - q_{i-1}t_{i-1}r_1 \pmod{r_0} \\ &\equiv (t_{i-2} - q_{i-1}t_{i-1})r_1 \pmod{r_0} \\ &\equiv t_ir_1 \pmod{r_0}. \end{aligned}$$

Tak więc z zasady indukcji wynika prawdziwość twierdzenia. ■

Następny wniosek jest bezpośrednią konsekwencją twierdzenia.

#### WNIOSEK 4.2

Jeśli  $\text{NWD}(r_0, r_1) = 1$ , to  $t_m = r_1^{-1} \pmod{r_0}$ .

Liczby z ciągu  $t_0, t_1, \dots, t_m$  można otrzymać w algorytmie Euklidesa jednocześnie z liczbami  $q_j$  i  $r_j$ . Na rysunku 4.1 przedstawiamy rozszerzony algorytm Euklidesa, umożliwiający obliczenie odwrotności liczby  $b$  modulo  $n$ , jeśli istnieje. W tej wersji algorytmu nie przechowujemy obliczonych już liczb  $q_j$ ,  $r_j$  i  $t_j$  w macierzy, gdyż na dowolnym etapie algorytmu wystarczy pamiętać tylko „ostatnie” dwa wyrazy każdego z tych ciągów.

W kroku 10 wyrażenie *temp* zapisaliśmy w taki sposób, by przy redukcji modulo  $n$  mieć do czynienia z argumentem dodatnim. (Wspominaliśmy już, że w wielu językach programowania redukcja tego typu daje dla liczb ujemnych także wyniki ujemne, podczas gdy tu chcielibyśmy na koniec obliczeń uzyskać wynik dodatni). Zauważmy też, że w kroku 12 zawsze jest spełniony warunek  $tb \equiv r \pmod{n}$  (co udowodniliśmy w twierdzeniu 4.1).

Oto mały przykład dla ilustracji.

#### Przykład 4.1

Przyjmijmy, że chcemy obliczyć  $28^{-1} \pmod{75}$ . Rozszerzony algorytm Euklidesa wykona następujące działania:

$75 = 2 \cdot 28 + 19$	krok 6
$73 \cdot 28 \pmod{75} = 19$	krok 12
$28 = 1 \cdot 19 + 9$	krok 16
$3 \cdot 28 \pmod{75} = 9$	krok 12
$19 = 2 \cdot 9 + 1$	krok 16
$67 \cdot 28 \pmod{75} = 1$	krok 12
$9 = 9 \cdot 1$	krok 16

W wyniku otrzymamy  $28^{-1} \pmod{75} = 67$ . □

```

(1)  $n_0 = n$ 
(2)  $b_0 = b$ 
(3)  $t_0 = 0$ 
(4)  $t = 1$ 
(5)  $q = \lfloor \frac{n_0}{b_0} \rfloor$ 
(6)  $r = n_0 - q \cdot b_0$ 
(7) while  $r > 0$  do
(8)    $temp = t_0 - q \cdot t$ 
(9)   if  $temp \geq 0$  then  $temp = temp \bmod n$ 
(10)  if  $temp < 0$  then  $temp = n - ((-temp) \bmod n)$ 
(11)   $t_0 = t$ 
(12)   $t = temp$ 
(13)   $n_0 = b_0$ 
(14)   $b_0 = r$ 
(15)   $q = \lfloor \frac{n_0}{b_0} \rfloor$ 
(16)   $r = n_0 - q \cdot b_0$ 
(17) if  $b_0 \neq 1$  then
       $b$  nie ma odwrotności modulo  $n$ 
    else
       $b^{-1} = t \bmod n$ 

```

RYSUNEK 4.1. Rozszerzony algorytm Euklidesa

#### 4.2.2. Chińskie twierdzenie o resztach

Chińskie twierdzenie o resztach jest w istocie metodą rozwiązywania pewnych układów kongruencji. Niech  $m_1, \dots, m_r$  będzie ciągiem parami względnie pierwszych liczb całkowitych dodatnich (czyli  $\text{NWD}(m_i, m_j) = 1$ , gdy  $i \neq j$ ) i niech  $a_1, \dots, a_r$  będą liczbami całkowitymi. Rozważmy następujący układ kongruencji:

$$x \equiv a_1 \pmod{m_1}$$

$$x \equiv a_2 \pmod{m_2}$$

 $\vdots$ 

$$x \equiv a_r \pmod{m_r}.$$

Chińskie twierdzenie o resztach stwierdza istnienie jednoznacznego rozwiązania modulo  $M = m_1 \cdot m_2 \cdot \dots \cdot m_r$ . W niniejszym podrozdziale udowodnimy ten wynik i opiszemy efektywny algorytm rozwiązywania układów kongruencji tego typu.

Warto w tym celu zająć się funkcją  $\pi: \mathbb{Z}_M \rightarrow \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \dots \times \mathbb{Z}_{m_r}$ , którą zdefiniujemy w sposób następujący:

$$\pi(x) = (x \bmod m_1, \dots, x \bmod m_r).$$

**Przykład 4.2**

Niech  $r = 2$ ,  $m_1 = 5$  i  $m_2 = 3$ ; wówczas  $M = 15$ . Funkcja  $\pi$  przyjmuje następujące wartości:

$$\begin{array}{lll} \pi(0) = (0, 0) & \pi(1) = (1, 1) & \pi(2) = (2, 2) \\ \pi(3) = (3, 0) & \pi(4) = (4, 1) & \pi(5) = (0, 2) \\ \pi(6) = (1, 0) & \pi(7) = (2, 1) & \pi(8) = (3, 2) \\ \pi(9) = (4, 0) & \pi(10) = (0, 1) & \pi(11) = (1, 2) \\ \pi(12) = (2, 0) & \pi(13) = (3, 1) & \pi(14) = (4, 2) \end{array}$$

□

Dowód chińskiego twierdzenia o resztach sprowadza się do wykazania, że tak określona funkcja  $\pi$  jest bijekcją. Łatwo zauważyc, że w przykładzie 4.2 tak właśnie jest. Teraz znajdziemy jawnego wzoru ogólnego dla funkcji odwrotnej  $\pi^{-1}$ .

Dla  $1 \leq i \leq r$  definiujemy

$$M_i = \frac{M}{m_i}.$$

Nietrudno zauważyc, że

$$\text{NWD}(M_i, m_i) = 1$$

dla  $1 \leq i \leq r$ . Dalej, dla  $1 \leq i \leq r$  definiujemy

$$y_i = M_i^{-1} \pmod{m_i}.$$

(Przypomnijmy, że ten element odwrotny istnieje, gdyż  $\text{NWD}(M_i, m_i) = 1$ , i można go znaleźć za pomocą algorytmu Euklidesa). Zauważmy, że

$$M_i y_i \equiv 1 \pmod{m_i}$$

dla  $1 \leq i \leq r$ .

Określmy teraz nową funkcję  $\rho : \mathbb{Z}_{m_1} \times \dots \times \mathbb{Z}_{m_r} \rightarrow \mathbb{Z}_M$  w następujący sposób:

$$\rho(a_1, \dots, a_r) = \sum_{i=1}^r a_i M_i y_i \pmod{M}.$$

Wykażemy, że  $\rho = \pi^{-1}$ , a więc że funkcja  $\rho$  stanowi jawnego wzoru rozwiązania pierwotnego układu kongruencji.

Przyjmijmy  $X = \rho(a_1, \dots, a_r)$  i niech  $1 \leq j \leq r$ . Rozważmy składnik  $a_i M_i y_i$  sumy występującej w definicji funkcji  $\rho$ , zredukowany modulo  $m_j$ . Jeśli  $i = j$ , to

$$a_i M_i y_i \equiv a_i \pmod{m_i},$$

ponieważ

$$M_i y_i \equiv 1 \pmod{m_i}.$$

Jednak gdy  $i \neq j$ ,

$$a_i M_i y_i \equiv 0 \pmod{m_j},$$

jako że w tym przypadku  $m_j | M_j$ . Mamy zatem

$$X \equiv \sum_{i=1}^r a_i M_i y_i \pmod{m_j} \equiv a_j \pmod{m_j}.$$

Ponieważ warunek ten jest spełniony dla każdego  $j$ , takiego że  $1 \leq j \leq r$ ,  $X$  stanowi rozwiązanie układu kongruencji.

Teraz musimy jeszcze wykazać, że rozwiązanie  $X$  jest jedyne modulo  $M$ , a to wymaga już tylko prostego przeliczenia. Dziedzina funkcji  $\pi$  jest zbiorem  $M$ -elementowym, także jej zbiór wartości ma  $M$  elementów. Dopiero co udowodniliśmy, że  $\pi$  jest funkcją surjektyną (czyli „na”). Wynika stąd, że jest także injektyną (czyli różnowartościowa), ponieważ dziedzina i zbiór wartości mają tyle samo elementów. Tak więc  $\pi$  jest bijekcją i  $\pi^{-1} = \rho$ . Dodajmy, że  $\pi^{-1}$  jest funkcją liniową swoich argumentów  $a_1, \dots, a_r$ .

Oto, dla ilustracji, przykład nieco obszerniejszy.

### Przykład 4.3

Niech  $r = 3$ ,  $m_1 = 7$ ,  $m_2 = 11$  oraz  $m_3 = 13$ ; wówczas  $M = 1001$ . Obliczamy  $M_1 = 143$ ,  $M_2 = 91$  i  $M_3 = 77$ , a następnie  $y_1 = 5$ ,  $y_2 = 4$  oraz  $y_3 = 12$ . Funkcja  $\pi^{-1} : \mathbb{Z}_7 \times \mathbb{Z}_{11} \times \mathbb{Z}_{13} \rightarrow \mathbb{Z}_{1001}$  wygląda wtedy tak:

$$\pi^{-1}(a_1, a_2, a_3) = 715a_1 + 364a_2 + 924a_3 \pmod{1001}.$$

Na przykład, jeśli  $x \equiv 5 \pmod{7}$ ,  $x \equiv 3 \pmod{11}$  oraz  $x \equiv 10 \pmod{13}$ , to ze wzoru otrzymujemy

$$\begin{aligned} x &= 715 \cdot 5 + 364 \cdot 3 + 924 \cdot 10 \pmod{1001} \\ &= 13907 \pmod{1001} \\ &= 894 \pmod{1001} \end{aligned}$$

Dla sprawdzenia wystarczy zredukować 894 modulo 7, 11 i 13. □

Sformułujmy wyniki przedstawione w tym podrozdziale w postaci twierdzenia, aby móc się na nie w przyszłości powoływać.

### **TWIERDZENIE 4.3** (chińskie twierdzenie o resztach)

Niech  $m_1, \dots, m_r$  będą parami względnie pierwszymi dodatnimi liczbami całkowitymi, a  $a_1, \dots, a_r$  niech będą liczbami całkowitymi. Wówczas układ  $r$  kongruencji  $x \equiv a_i \pmod{m_i}$  ( $1 \leq i \leq r$ ) ma jednoznaczne rozwiązanie modulo  $M = m_1 \cdot \dots \cdot m_r$  dane wzorem

$$x = \sum_{i=1}^r a_i M_i y_i \pmod{M},$$

gdzie  $M_i = M/m_i$  oraz  $y_i = M_i^{-1} \pmod{m_i}$ , dla  $1 \leq i \leq r$ .

### 4.2.3. Dalsze pożyteczne fakty

Następny wynik pochodzi z elementarnej teorii grup. Nosi nazwę twierdzenia Lagrange'a i będzie miał szczególne znaczenie przy omawianiu systemu kryptograficznego RSA. Niech  $G$  będzie (skończoną) grupą mnożyciową. Rzędem elementu  $g \in G$  nazywiemy najmniejszą liczbę całkowitą  $m$ , taką że  $g^m = 1$ . Wynik, który przedstawiamy, jest dość prosty, ale nie będziemy go tu dowodzić.

#### TWIERDZENIE 4.4 (Lagrange'a)

Niech  $G$  będzie grupą mnożyciową rzędu  $n$  oraz niech  $g \in G$ . Wówczas rząd elementu  $g$  dzieli  $n$ .

Zasadniczą rolę w naszych dalszych rozważaniach odegrają następujące dwa wnioski.

#### WNIOSEK 4.5

Jeśli  $b \in \mathbb{Z}_n^*$ , to  $b^{\phi(n)} \equiv 1 \pmod{n}$ .

**DOWÓD**  $\mathbb{Z}_n^*$  jest grupą mnożyciową rzędu  $\phi(n)$ . ■

#### WNIOSEK 4.6 (Fermata)

Jeśli  $p$  jest liczbą pierwszą i  $b \in \mathbb{Z}_p$ , to  $b^p \equiv b \pmod{p}$ .

**DOWÓD** Jeśli  $p$  jest liczbą pierwszą, to  $\phi(p) = p - 1$ . Zatem gdy  $b \not\equiv 0 \pmod{p}$ , teza wynika z wniosku 4.5. Gdy  $b \equiv 0 \pmod{p}$ , teza pozostaje prawdziwa, gdyż  $0^p \equiv 0 \pmod{p}$ . ■

Wiemy zatem, że gdy  $p$  jest liczbą pierwszą, rząd grupy  $\mathbb{Z}_p^*$  jest równy  $p - 1$ , a rząd każdego elementu tej grupy jest dzielnikiem liczby  $p - 1$ . W rzeczywistości, gdy  $p$  jest liczbą pierwszą, grupa  $\mathbb{Z}_p^*$  jest nawet cykliczna: istnieje element  $\alpha \in \mathbb{Z}_p^*$ , którego rząd jest także równy  $p - 1$ . Nie będziemy dowodzić tego bardzo ważnego faktu, chcemy go tu jedynie odnotować gwoli przyszłego wykorzystania.

#### TWIERDZENIE 4.7

Jeśli  $p$  jest liczbą pierwszą, to grupa  $\mathbb{Z}_p^*$  jest cykliczna.

Element  $\alpha$  rzędu  $p - 1$  nazywamy elementem *pierwotnym* modulo  $p$ . Zauważmy, że  $\alpha$  jest elementem pierwotnym wtedy i tylko wtedy, gdy

$$\{\alpha^i : 0 \leq i \leq p - 2\} = \mathbb{Z}_p^*.$$

Załóżmy, że  $p$  jest liczbą pierwszą, a  $\alpha$  jest elementem pierwotnym modulo  $p$ . Każdy element  $\beta \in \mathbb{Z}_p^*$  możemy zapisać jednoznacznie w postaci  $\beta = \alpha^i$ , gdzie  $0 \leq i \leq p - 2$ . Nietrudno wykazać, że rząd elementu  $\beta$  jest równy

$$\frac{p - 1}{\text{NWD}(p - 1, i)}.$$

Tak więc  $\beta$  jest elementem pierwotnym wtedy i tylko wtedy, gdy  $\text{NWD}(p-1, i) = 1$ . Wynika stąd, że liczba elementów pierwotnych modulo  $p$  jest równa  $\phi(p-1)$ .

#### Przykład 4.4

Niech  $p = 13$ . Obliczając kolejne potęgi liczby 2, możemy sprawdzić, że 2 jest elementem pierwotnym modulo 13.

$$2^0 \bmod 13 = 1$$

$$\underline{2^1 \bmod 13 = 2}$$

$$2^2 \bmod 13 = 4$$

$$2^3 \bmod 13 = 8$$

$$2^4 \bmod 13 = 3$$

$$\underline{2^5 \bmod 13 = 6}$$

$$2^6 \bmod 13 = 12$$

$$\underline{2^7 \bmod 13 = 11}$$

$$2^8 \bmod 13 = 9$$

$$2^9 \bmod 13 = 5$$

$$2^{10} \bmod 13 = 10$$

$$\underline{2^{11} \bmod 13 = 7}$$

Element  $2^i$  jest pierwotny wtedy i tylko wtedy, gdy  $\text{NWD}(i, 12) = 1$ , a więc wtedy i tylko wtedy, gdy  $i = 1, 5, 7$  lub  $11$ . Dlatego elementami pierwotnymi modulo 13 są: 2, 6, 7 oraz 11.  $\square$

### 4.3. System kryptograficzny RSA

Opiszymy teraz system kryptograficzny RSA. Wszystkie obliczenia w tym kryptosystemie wykonuje się w  $\mathbb{Z}_n$ , gdzie  $n$  jest iloczynem dwóch różnych liczb pierwszych nieparzystych  $p$  i  $q$ . Warto zauważyć, że dla takiej liczby  $n$ ,  $\phi(n) = (p-1)(q-1)$ .

Formalny opis systemu przedstawiamy na rysunku 4.2. Przekonajmy się o tym, że szyfrowanie i deszyfrowanie są istotnie operacjami wzajemnie do siebie odwrotnymi. Ponieważ

$$ab \equiv 1 \pmod{\phi(n)},$$

to

$$ab = t\phi(n) + 1$$

Niech  $n = pq$  dla liczb pierwszych  $p$  i  $q$  oraz  $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$ . Określmy

$$\mathcal{K} = \{(n, p, q, a, b) : n = pq, p, q \text{ pierwsze}, ab \equiv 1 \pmod{\phi(n)}\}.$$

Dla  $K = (n, p, q, a, b)$  definiujemy

$$e_K(x) = x^b \pmod{n}$$

oraz

$$d_K(y) = y^a \pmod{n}$$

$(x, y \in \mathbb{Z}_n)$ . Wartości  $n$  i  $b$  są znane publicznie, a liczby  $p, q, a$  są tajne.

#### RYSUNEK 4.2. System kryptograficzny RSA

dla pewnej liczby całkowitej  $t \geq 1$ . Niech  $x \in \mathbb{Z}_n^*$ ; wówczas

$$\begin{aligned} (x^b)^a &\equiv x^{t\phi(n)+1} \pmod{n} \\ &\equiv (x^{\phi(n)})^t x \pmod{n} \\ &\equiv 1^t x \pmod{n} \\ &\equiv x \pmod{n}, \end{aligned}$$

tak jak się spodziewaliśmy. Pozostawiamy Czytelnikowi jako ćwiczenie wykazać, że  $(x^b)^a \equiv x \pmod{n}$ , gdy  $x \in \mathbb{Z}_n \setminus \mathbb{Z}_n^*$ .

Oto uproszczony (i niezbyt bezpieczny) przykład systemu kryptograficznego RSA.

#### Przykład 4.5

Przyjmijmy, że Bolek wybrał  $p = 101$  i  $q = 113$ . Wtedy  $n = 11413$  oraz  $\phi(n) = 100 \cdot 112 = 11200$ . Ponieważ  $11200 = 2^6 5^2 7$ , liczba całkowita  $b$  może posłużyć za wykładnik reguły szyfrowania wtedy i tylko wtedy, gdy nie jest podzielna przez żadną z liczb 2, 5 i 7. (W praktyce Bolek nie będzie rozkładał liczby  $\phi(n)$  na czynniki. Sprawdzi jedynie za pomocą algorytmu Euklidesa, czy  $\text{NWD}(\phi(n), b) = 1$ ). Założymy, że Bolek przyjął  $b = 3533$ . Wówczas z rozszerzonego algorytmu Euklidesa otrzymujemy

$$b^{-1} = 6597 \pmod{11200}.$$

Tak więc tajnym wykładnikiem deszyfrowania jest  $a = 6597$ .

Bolek publikuje liczby  $n = 11413$  i  $b = 3533$  w informatorze. Przypuśćmy teraz, że Alicja chce przesłać Bolekowi tekst jawnego 9726. Oblicza w tym celu:

$$9726^{3533} \pmod{11413} = 5761,$$

po czym wysyła przez odpowiedni kanał tekst zaszyfrowany 5761. Gdy Bolek go otrzymuje, używa swojego tajnego wykładnika deszyfrowania, by obliczyć:

$$5761^{6597} \text{ mod } 11413 = 9726.$$

(W tej chwili operacje szyfrowania i deszyfrowania mogą wydawać się bardzo skomplikowane, ale w następnym podrozdziale zajmiemy się efektywnymi algorytmami, które je wykonują).  $\square$

Bezpieczeństwo systemu RSA opiera się na założeniu, że funkcja szyfrująca  $e_K(x) = x^b \text{ mod } n$  jest jednokierunkowa, a więc zadanie deszyfrowania tekstu zaszyfrowanego jest dla przeciwnika obliczeniowo niewykonalne. Bocznym wyjściem umożliwiającym Bolkowi odczytanie komunikatu jest jego znajomość rozkładu  $n = pq$ . Potrafi on dzięki temu obliczyć  $\phi(n) = (p-1)(q-1)$ , a następnie za pomocą algorytmu Euklidesa także wykładnik deszyfrowania  $a$ . W dalszej części rozdziału poświęcimy więcej miejsca kwestii bezpieczeństwa kryptosystemu RSA.

#### 4.4. Implementacja RSA

Przy omawianiu systemu kryptograficznego RSA należy zwrócić uwagę na wiele jego aspektów: szczegóły budowania systemu, efektywność szyfrowania i deszyfrowania, sprawy bezpieczeństwa. Budując system, Bolek postępuje tak, jak to jest przedstawione na rysunku 4.3. O tym, jak wykonuje opisane tam działania, dowiemy się w dalszej części rozdziału.

- (1) Bolek generuje dwie liczby pierwsze,  $p$  i  $q$
- (2) Bolek oblicza  $n = pq$  i  $\phi(n) = (p-1)(q-1)$
- (3) Bolek wybiera losowo  $b$  ( $1 < b < \phi(n)$ ), takie że  $\text{NWD}(b, \phi(n)) = 1$
- (4) za pomocą algorytmu Euklidesa Bolek oblicza  $a = b^{-1} \text{ mod } \phi(n)$
- (5) Bolek publikuje w informatorze liczby  $n$  i  $b$  jako swój klucz publiczny

RYSUNEK 4.3. Konstrukcja RSA

Jednym z oczywistych sposobów ataku na taki system kryptograficzny jest próba rozłożenia liczby  $n$  na czynniki. Jeśli da się to zrobić, to obliczenie  $\phi(n) = (p-1)(q-1)$  jest już rzeczą prostą, po której można obliczyć wykładnik deszyfrowania  $a$  na podstawie znajomości  $b$  dokładnie tak, jak to zrobił Bolek.

(Istnieje hipoteza, że złamanie szyfru RSA jest wielomianowo równoważne<sup>2</sup> faktoryzacji liczby  $n$ , choć dotąd nie udało się tego udowodnić).

Tak więc, jeśli system RSA ma być bezpieczny, liczba  $n = pq$  musi być dostatecznie duża, by faktoryzacja okazała się obliczeniowo niewykonalna. Za pomocą używanych obecnie algorytmów rozkładanie na czynniki można zrobić dla liczb, które mają do 130 cyfr dziesiętnych (więcej informacji o faktoryzacji w podrozdziale 4.8). Stąd zaleca się dla bezpieczeństwa, by każda z liczb pierwszych  $p$  i  $q$  miała co najmniej 100 cyfr w zapisie dziesiętnym; wtedy  $n$  będzie liczbą 200-cyfrową. Wiele sprzętowych implementacji systemu RSA używa modułów 512-bitowych, jednak taki moduł odpowiada ok. 154 cyfrom dziesiętnym (ponieważ liczba bitów w reprezentacji dziesiętnej liczby całkowitej jest  $\log_2 10$  razy większa od liczby cyfr dziesiętnych), a to za mało, by zapewnić wystarczające bezpieczeństwo na dłuższą metę.

Pozostawiając chwilowo na boku pytanie, jak znaleźć dwie 100-cyfrowe liczby pierwsze, zatrzymajmy się przy arytmetycznych operacjach szyfrowania i deszyfrowania. Szyfrowanie (lub deszyfrowanie) wymaga jednego potęgowania modulo  $n$ . Zważywszy na wielkość liczby  $n$ , musimy do wykonania działań w  $\mathbb{Z}_n$  użyć arytmetyki wielokrotnej precyzji, a czas potrzebny do tego będzie zależał od liczby bitów występujących w binarnej reprezentacji liczby  $n$ .

Załóżmy, że  $n$  ma  $k$ -bitową reprezentację binarną, a więc  $k = \lfloor \log_2 n \rfloor + 1$ . Nietrudno stwierdzić, wykorzystując do tego celu standardowe metody arytmetyczne rodem ze szkoły podstawowej, że dodawanie dwóch  $k$ -bitowych liczb całkowitych można wykonać w czasie  $O(k)$ , mnożenie zaś w czasie  $O(k^2)$ . Podobnie w czasie  $O(k^2)$  można zredukować modulo  $n$  liczbę całkowitą co najwyżej  $2k$ -bitową (jako że operacja ta sprowadza się do wykonania dzielenia i zachowania reszty). Niech  $x, y \in \mathbb{Z}_n$  (zakładamy przy tym, że  $0 \leq x, y \leq n - 1$ ). Wtedy  $xy \bmod n$  można obliczyć, mnożąc najpierw liczby  $x$  i  $y$  (otrzymuje się wtedy liczbę  $2k$ -bitową), a następnie redukując iloczyn modulo  $n$ . Oba te kroki można wykonać w czasie  $O(k^2)$ . Takie obliczenie nazwiemy *mnożeniem modularnym*.

Przyjrzyjmy się teraz *potęgowaniu modularnemu*, a więc obliczaniu funkcji postaci  $x^c \bmod n$ . Jak stwierdziliśmy wyżej, zarówno szyfrowanie, jak i deszyfrowanie w RSA wymagają potęgowania modularnego. Liczbę  $x^c \bmod n$  da się obliczyć za pomocą  $c - 1$  mnożeń modularnych, ale ten sposób jest wysoce nieefektywny, gdy liczba  $c$  jest duża – a zauważmy, że  $c$  może być tak duża jak liczba  $\phi(n) - 1$ , wykładniczo duża w porównaniu z  $k$ .

Dobrze znana metoda „podnieś do kwadratu i wymnóż” umożliwia zmniejszenie liczby mnożeń modularnych potrzebnych do obliczenia  $x^c \bmod n$  do co najwyżej  $2\ell$ , gdzie  $\ell$  jest liczbą bitów w binarnym zapisie liczby  $c$ . Nierówność  $\ell \leq k$  oznacza, że do obliczenia  $x^c \bmod n$  wystarczy czas  $O(k^3)$ . Dlatego zarówno szyfrowanie, jak i deszyfrowanie w systemie RSA można wykonać w czasie wielomianowym (będącym funkcją  $k$ , liczby bitów w jednym znaku tekstu jawnego (lub zaszyfrowanego)).

<sup>2</sup>Dwa problemy są *wielomianowo równoważne*, gdy istnienie algorytmu rozwiązującego którykolwiek z nich w czasie wielomianowym pociąga za sobą istnienie takiego algorytmu dla drugiego problemu.

(1)	$z = 1$
(2)	<b>for</b> $i = \ell - 1$ <b>downto</b> 0 <b>do</b>
(3)	$z = z^2 \bmod n$
(4)	<b>if</b> $b_i = 1$ <b>then</b> $z = z \cdot x \bmod n$

RYSUNEK 4.4. Algorytm podnoszenia do kwadratu i wymnażania obliczający  $x^b \bmod n$

Podnoszenie do kwadratu i mnożenie opiera się na założeniu, że wykładnik, powiedzmy  $b$ , jest dany w zapisie binarnym:

$$b = \sum_{i=0}^{\ell-1} b_i 2^i,$$

gdzie  $b_i = 0$  lub  $1$ ,  $0 \leq i \leq \ell - 1$ . Na rysunku 4.4 prezentujemy algorytm obliczania  $z = x^b \bmod n$ . Łatwo policzyć, ile ten algorytm wykonuje mnożeń modularnych. Zawsze podnosimy  $\ell$  razy do kwadratu (krok 3). Liczba mnożeń modularnych w kroku 4 jest równa liczbie jedynek w binarnym zapisie liczby  $b$ , a więc pewnej liczbie całkowitej z przedziału od 0 do  $\ell$ . W rezultacie łączna liczba modularnych mnożeń jest nie mniejsza od  $\ell$  i nie większa od  $2\ell$ .

Wróćmy do przykładu 4.5, by zobaczyć działanie algorytmu.

#### Przykład 4.5 cd.

Przypomnijmy, że  $n = 11413$ , a publicznym wykładniakiem szyfrowania jest  $b = 3533$ . Alicja szyfruje tekst jawny 9726, obliczając za pomocą algorytmu podnoszenia do kwadratu i mnożenia liczbę  $9726^{3533} \bmod 11413$ :

$i$	$b_i$	$z$
11	1	$1^2 \cdot 9726 = 9726$
10	1	$9726^2 \cdot 9726 = 2659$
9	0	$2659^2 = 5634$
8	1	$5634^2 \cdot 9726 = 9167$
7	1	$9167^2 \cdot 9726 = 4958$
6	1	$4958^2 \cdot 9726 = 7783$
5	0	$7783^2 = 6298$
4	0	$6298^2 = 4629$
3	1	$4629^2 \cdot 9726 = 10185$
2	1	$10185^2 \cdot 9726 = 105$
1	0	$105^2 = 11025$
0	1	$11025^2 \cdot 9726 = 5761$

Tak więc, jak stwierdziliśmy wcześniej, tekstem zaszyfrowanym jest 5761.  $\square$

Należy podkreślić, że obecnie najbardziej efektywne implementacje systemu RSA osiągają szybkość szyfrowania rzędu 600 kbitów na sekundę (przy użyciu

512-bitowego modułu  $n$ ), podczas gdy dla DES osiąga się 1 Gbit na sekundę. Mówiąc inaczej, RSA jest około 1500 razy wolniejszy od DES.

Omówiliśmy dotąd operacje szyfrowania i deszyfrowania w kryptosystemie RSA. W następnym podrozdziale zajmiemy się generowaniem liczb pierwszych  $p$  i  $q$  (krok 1). Krok 2 – bezpośrednie obliczenia – można wykonać w czasie  $O((\log n)^2)$ . W krokach 3 i 4 stosuje się algorytm Euklidesa, więc rozważmy tu krótko ich złożoność.

Załóżmy, że obliczamy największy wspólny dzielnik liczb  $r_0$  i  $r_1$ , przy czym  $r_0 > r_1$ . W każdej iteracji algorytmu obliczamy iloraz i resztę, co można uczynić w czasie  $O((\log r_0)^2)$ . Jeśli potrafimy określić górne ograniczenie liczby iteracji, to otrzymamy zarazem ograniczenie złożoności algorytmu. Takie ograniczenie uzyskamy dzięki twierdzeniu Lamégo. Twierdzenie głosi, że jeśli  $s$  jest liczbą iteracji algorytmu, to  $f_{s+2} \leq r_0$ , gdzie  $f_i$  oznacza  $i$ -tą liczbę Fibonacciego. Ponieważ

$$f_i \approx \left( \frac{1 + \sqrt{5}}{2} \right)^i,$$

wnioskujemy, że  $s$  jest wielkością rzędu  $O(\log r_0)$ .

Wynika stąd, że czas przebiegu algorytmu Euklidesa wynosi  $O((\log n)^3)$ . Wykonując bardziej wnikliwą analizę, stwierdzimy, że jest on równy  $O((\log n)^2)$ .

#### 4.5. Probabilistyczny test na liczby pierwsze

Przy konstruowaniu systemu kryptograficznego RSA są potrzebne dwie duże (np. 80-cyfrowe) „losowe liczby pierwsze”. W praktyce generuje się duże liczby losowe, po czym sprawdza się, czy są to liczby pierwsze, za pomocą probabilistycznego algorytmu typu Monte Carlo działającego w czasie wielomianowym, takiego jak algorytmy Solovaya–Strassena lub Millera–Rabina. Oba omówimy w tym podrozdziale. Są to szybkie algorytmy (liczbę  $n$  sprawdza się w czasie wielomianowym ze względu na  $\log_2 n$ , liczbę bitów w binarnym zapisie liczby  $n$ ), ale nie można wykluczyć, że algorytm rozpozna jako liczbę pierwszą liczbę, która taka nie jest. Można jednak zmniejszyć to prawdopodobieństwo – poniżej dowolnego ustalonego progu – uruchamiając algorytm wystarczającą liczbę razy. Nieco dalej zajmiemy się tym tematem bardziej szczegółowo.

Drugą ważną kwestią jest liczba liczb losowych (określonej wielkości), które trzeba sprawdzić, by znaleźć liczbę pierwszą. Znane twierdzenie teorii liczb, zwane twierdzeniem o liczbach pierwszych, mówi, że wśród pierwsiych  $N$  liczb naturalnych jest około  $N / \ln N$  liczb pierwszych. W związku z tym, jeśli wybieramy losowo liczbę  $p$ , prawdopodobieństwo, że będzie to liczba pierwsza, wynosi około  $1 / \ln p$ . Dla 512-bitowego modułu mamy  $1 / \ln p \approx 1 / 177$ . Oznacza to, że średnio spośród 177 losowo wybranych liczb całkowitych  $p$  odpowiedniej wielkości jedna okaza się pierwsza (oczywiście jeśli ograniczymy się do poszukiwania

wśród liczb nieparzystych, prawdopodobieństwo się podwoi i będzie równe około  $2/177$ ). Tak więc generowanie dostatecznie dużych liczb losowych „prawdopodobnie pierwszych” jest rzeczywiście wykonalne, a zatem i konstrukcja systemu RSA jest realna. Opiszemy teraz, jak to się robi.

*Problem decyzyjny* to problem, w którym odpowiedzią na postawione w nim pytanie jest „tak” albo „nie”. Algorytm nazwiemy *probabilistycznym*, jeśli wykorzystuje liczby losowe (pozostałe algorytmy, a więc te, które liczb losowych nie używają, nazwiemy *deterministycznymi*). Następna definicja odnosi się do probabilistycznych algorytmów rozstrzygania problemów decyzyjnych.

#### DEFINICJA 4.1

*Pozytywnie nastawionym algorytmem Monte Carlo* nazywamy algorytm probabilistyczny rozstrzygania problemu decyzyjnego, w którym odpowiedź „tak” jest zawsze poprawna, natomiast odpowiedź „nie” może być błędna. *Negatywnie nastawiony algorytm Monte Carlo* definiujemy analogicznie. Mówimy, że pozytywnie nastawiony algorytm Monte Carlo ma *prawdopodobieństwo błędu* równe  $\epsilon$ , jeśli w każdym przypadku, gdy poprawną odpowiedzią jest „tak”, algorytm poda (niepoprawną) odpowiedź „nie” z prawdopodobieństwem co najwyżej  $\epsilon$ . (To prawdopodobieństwo oblicza się w stosunku do wszystkich możliwych losowych wyborów dokonywanych przez algorytm działający na ustalonych danych wejściowych).

Na rysunku 4.5 przedstawiamy problem decyzyjny **rozkładalność**.

**Dane** Dodatnia liczba całkowita  $n \geq 2$ .

**Pytanie** Czy  $n$  jest złożona?

#### RYSUNEK 4.5. Rozkładalność

Przypomnijmy, że algorytm dla problemu decyzyjnego ma jedynie udzielić odpowiedzi „tak” lub „nie”. W szczególności w przypadku problemu rozkładalność nie oczekujemy od algorytmu wskazania rozkładu, jeśli liczba  $n$  jest złożona.

Najpierw omówimy algorytm Solovaya–Strassena, pozytywnie nastawiony algorytm Monte Carlo dla problemu rozkładalności z prawdopodobieństwem błędu równym  $1/2$ . Oznacza to, że gdy algorytm odpowiada „tak”, liczba  $n$  jest złożona; i na odwrót, jeśli  $n$  jest złożona, algorytm odpowie „tak” z prawdopodobieństwem  $1/2$ .

Algorytm Millera–Rabina (którym zajmiemy się później) jest co prawda szybszy od algorytmu Solovaya–Strassena, zaczniemy jednak od tego ostatniego, gdyż jest on pojęciowo prostszy, a ponadto pojawiają się w nim pewne pojęcia teorioliczbowe, które przydadzą się w dalszych rozdziałach książki. Zanim przejdziemy do opisu algorytmu, musimy nieco wzbogacić zasób instrumentów zaczerpniętych z teorii liczb.

**DEFINICJA 4.2**

Niech  $p$  będzie nieparzystą liczbą pierwszą, a  $x$  liczbą całkowitą, przy czym  $1 \leq x \leq p - 1$ . Mówimy, że  $x$  jest resztą kwadratową modulo  $p$ , gdy kongruencja  $y^2 \equiv x \pmod{p}$  ma rozwiązanie  $y \in \mathbb{Z}_p$ . Element  $x$  nazywamy nieresztą kwadratową modulo  $p$ , gdy  $x \not\equiv 0 \pmod{p}$  oraz  $x$  nie jest resztą kwadratową modulo  $p$ .

**Przykład 4.6**

Resztami kwadratowymi modulo 11 są liczby 1, 3, 4, 5 i 9, ponieważ  $(\pm 1)^2 = 1$ ,  $(\pm 5)^2 = 3$ ,  $(\pm 2)^2 = 4$ ,  $(\pm 4)^2 = 5$  oraz  $(\pm 3)^2 = 9$  (wszystkie działania wykonujemy tu w  $\mathbb{Z}_{11}$ ).  $\square$

Na rysunku 4.6 definiujemy w oczywisty sposób problem decyzyjny **reszty kwadratowe**.

**Dane** Nieparzysta liczba pierwsza  $p$  oraz liczba całkowita  $x$ , taka że  $0 \leq x \leq p - 1$ .

**Pytanie** Czy  $x$  jest resztą kwadratową modulo  $p$ ?

**RYSUNEK 4.6. Reszty kwadratowe**

Udowodnimy teraz wynik znany jako *kryterium Eulera*, który leży u podstaw deterministycznego algorytmu o czasie wielomianowym, rozstrzygającego problem reszt kwadratowych.

**TWIERDZENIE 4.8** (kryterium Eulera)

Niech  $p$  będzie nieparzystą liczbą pierwszą. Wówczas  $x$  jest resztą kwadratową modulo  $p$  wtedy i tylko wtedy, gdy

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

**DOWÓD** Załóżmy najpierw, że  $x \equiv y^2 \pmod{p}$ . Z wniosku 4.6 wiemy, że gdy  $p$  jest liczbą pierwszą, to  $x^{p-1} \equiv 1 \pmod{p}$  dla dowolnej liczby  $x$ , takiej że  $x \not\equiv 0 \pmod{p}$ . Mamy zatem

$$\begin{aligned} x^{(p-1)/2} &\equiv (y^2)^{(p-1)/2} \pmod{p} \\ &\equiv y^{p-1} \pmod{p} \\ &\equiv 1 \pmod{p}. \end{aligned}$$

Na odwrót załóżmy, że  $x^{(p-1)/2} \equiv 1 \pmod{p}$ . Niech  $b$  będzie elementem pierwotnym modulo  $p$ . Wtedy  $x \equiv b^i \pmod{p}$  dla pewnego  $i$ . Mamy wówczas

$$\begin{aligned} x^{(p-1)/2} &\equiv (b^i)^{(p-1)/2} \pmod{p} \\ &\equiv b^{i(p-1)/2} \pmod{p}. \end{aligned}$$

Rząd elementu  $b$  jest równy  $p - 1$ , zatem  $p - 1$  musi być dzielnikiem  $i(p - 1)/2$ , co oznacza, że  $i$  jest liczbą parzystą, a  $\pm b^{i/2}$  są pierwiastkami kwadratowymi liczby  $x$ . ■

Z twierdzenia 4.8 wynika pewien algorytm sprawdzania reszt kwadratowych o wielomianowym czasie działania, w którym stosuje się metodę podnoszenia do kwadratu i wymnażania do wykonania potęgowania modulo  $p$ . Algorytm ten ma złożoność  $O((\log p)^3)$ .

Potrzebujemy jeszcze kilku definicji z teorii liczb.

#### DEFINICJA 4.3

Niech  $p$  będzie nieparzystą liczbą pierwszą. Dla dowolnej liczby całkowitej  $a \geq 0$  definiujemy symbol Legendre'a  $\left(\frac{a}{p}\right)$  w sposób następujący:

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & \text{gdy } a \equiv 0 \pmod{p}, \\ 1, & \text{gdy } a \text{ jest resztą kwadratową modulo } p, \\ -1, & \text{gdy } a \text{ jest nieresztą kwadratową modulo } p. \end{cases}$$

Wiemy już, że  $a^{(p-1)/2} \equiv 1 \pmod{p}$  wtedy i tylko wtedy, gdy  $a$  jest resztą kwadratową modulo  $p$ . Jeśli  $a$  jest wielokrotnością  $p$ , to oczywiście  $a^{(p-1)/2} \equiv 0 \pmod{p}$ . Wreszcie, gdy  $a$  jest nieresztą kwadratową modulo  $p$ , to  $a^{(p-1)/2} \equiv -1 \pmod{p}$ , gdyż  $a^{p-1} \equiv 1 \pmod{p}$ . Mamy zatem następującą własność, która dostarcza efektywnego algorytmu obliczania symboli Legendre'a.

#### TWIERDZENIE 4.9

Niech  $p$  będzie nieparzystą liczbą pierwszą. Wówczas

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

Zdefiniujmy teraz pewne uogólnienie symbolu Legendre'a.

#### DEFINICJA 4.4

Niech  $p$  będzie nieparzystą liczbą całkowitą dodatnią i niech  $p_1^{e_1} \dots p_k^{e_k}$  będzie rozkładem liczby  $n$  na czynniki pierwsze. Dla liczby całkowitej  $a \geq 0$  definiujemy symbol Jacobiego  $\left(\frac{a}{n}\right)$  w następujący sposób:

$$\left(\frac{a}{n}\right) = \prod_{i=1}^k \left(\frac{a}{p_i}\right)^{e_i}.$$

### Przykład 4.7

Obliczymy symbol Jacobiego  $(\frac{6278}{9975})$ . Mamy następujący rozkład liczby 9975:  $9975 = 3 \cdot 5^2 \cdot 7 \cdot 19$ . Stąd

$$\begin{aligned} \left(\frac{6278}{9975}\right) &= \left(\frac{6278}{3}\right) \left(\frac{6278}{5}\right)^2 \left(\frac{6278}{7}\right) \left(\frac{6278}{19}\right) \\ &= \left(\frac{2}{3}\right) \left(\frac{3}{5}\right)^2 \left(\frac{6}{7}\right) \left(\frac{8}{19}\right) \\ &= (-1)(-1)^2(-1)(-1) \\ &= 1. \end{aligned}$$

□

Niech  $n > 1$  będzie liczbą nieparzystą. Jeśli  $n$  jest pierwsza, to  $(\frac{a}{n}) \equiv a^{(n-1)/2} \pmod{n}$  dla dowolnej liczby  $a$ . Z drugiej strony, jeśli  $n$  jest złożona, to może – ale nie musi! – być tak, że  $(\frac{a}{n}) \equiv a^{(n-1)/2} \pmod{n}$ . Jeśli tak jest, liczbę  $n$  nazwiemy *liczbą pseudopierwszą Eulera* przy podstawie  $a$ . Na przykład, 91 jest pseudopierwszą liczbą Eulera przy podstawie 10, ponieważ

$$\left(\frac{10}{91}\right) = -1 = 10^{45} \pmod{91}.$$

Można jednak wykazać, że dowolna nieparzysta liczba złożona  $n$  jest pseudopierwszą liczbą Eulera przy podstawie  $a$  dla co najwyżej połowy spośród liczb całkowitych  $a$ , takich że  $1 \leq a \leq n-1$  (patrz ćwiczenia). Ten fakt dowodzi, że algorytm Solovaya–Strassena sprawdzania pierwszości liczb, przedstawiony na rysunku 4.7, jest pozytywnie nastawionym algorymem Monte Carlo z prawdopodobieństwem błędu równym co najwyżej  $1/2$ . Na razie nie wiemy jeszcze, czy algorytm działa w czasie wielomianowym. Umiemy już obliczyć  $a^{(n-1)/2} \pmod{n}$  w czasie  $O((\log n)^3)$ , ale jak efektywnie obliczyć symbole Jacobiego? Mogliby się wydawać, że trzeba najpierw rozłożyćąć liczbę  $n$  na czynniki pierwsze, ponieważ czynniki te występują w definicji symbolu Jacobiego  $(\frac{a}{n})$ . Gdybyśmy jednak umieli rozłożyćąć  $n$ , wiedzielibyśmy już, czy  $n$  jest liczbą pierwszą czy nie, zatem takie podejście kończy się błędny kołem.

- (1) wybierz losowo liczbę całkowitą  $a$ ,  $1 \leq a \leq n-1$
- (2) if  $(\frac{a}{n}) \equiv a^{(n-1)/2} \pmod{n}$  then  
    odpowiedź „ $n$  jest pierwsza”  
else  
    odpowiedź „ $n$  jest złożona”

**RYSUNEK 4.7. Test pierwszości Solovaya–Strassena dla nieparzystej liczby całkowitej  $n$**

Na szczęście, możemy obliczyć symbol Jacobiego bez rozkładania na czynniki liczby  $n$ , korzystając za to z kilku wyników teorii liczb, z których najważniejszym

jest uogólnienie prawa kwadratowej wzajemności (własność 4 poniżej). Wymieńmy teraz te własności bez dowodu.

- (1) Jeśli  $n$  jest nieparzystą liczbą całkowitą oraz  $m_1 \equiv m_2 \pmod{n}$ , to

$$\left(\frac{m_1}{n}\right) = \left(\frac{m_2}{n}\right).$$

- (2) Jeśli  $n$  jest nieparzystą liczbą całkowitą, to

$$\left(\frac{2}{n}\right) = \begin{cases} 1, & \text{gdy } n \equiv \pm 1 \pmod{8}, \\ -1, & \text{gdy } n \equiv \pm 3 \pmod{8}. \end{cases}$$

- (3) Jeśli  $n$  jest nieparzystą liczbą całkowitą, to

$$\left(\frac{m_1 m_2}{n}\right) = \left(\frac{m_1}{n}\right) \left(\frac{m_2}{n}\right).$$

W szczególności, gdy  $m = 2^k t$  i  $t$  jest liczbą nieparzystą, to

$$\left(\frac{m}{n}\right) = \left(\frac{2}{n}\right)^k \left(\frac{t}{n}\right).$$

- (4) Niech  $m$  i  $n$  będą nieparzystymi liczbami całkowitymi. Wówczas

$$\left(\frac{m}{n}\right) = \begin{cases} -\left(\frac{n}{m}\right), & \text{gdy } m \equiv n \equiv 3 \pmod{4}, \\ \left(\frac{n}{m}\right), & \text{w przeciwnym przypadku.} \end{cases}$$

### Przykład 4.8

Aby zilustrować zastosowanie tych własności, obliczymy symbol Jacobiego  $\left(\frac{7411}{9283}\right)$  w następujący sposób:

$\left(\frac{7411}{9283}\right)$	$=$	$- \left(\frac{9283}{7411}\right)$	z własności 4
	$=$	$- \left(\frac{1872}{7411}\right)$	z własności 1
	$=$	$- \left(\frac{2}{7411}\right)^4 \left(\frac{117}{7411}\right)$	z własności 3
	$=$	$- \left(\frac{117}{7411}\right)$	z własności 2
	$=$	$- \left(\frac{7411}{117}\right)$	z własności 4

$$\begin{aligned}
 &= -\left(\frac{40}{117}\right) && \text{z własności 1} \\
 &= -\left(\frac{2}{117}\right)^3 \left(\frac{5}{117}\right) && \text{z własności 3} \\
 &= \left(\frac{5}{117}\right) && \text{z własności 2} \\
 &= \left(\frac{117}{5}\right) && \text{z własności 4} \\
 &= \left(\frac{2}{5}\right) && \text{z własności 1} \\
 &= -1 && \text{z własności 2}
 \end{aligned}$$

Zwróćmy uwagę na kolejne stosowanie w tym obliczeniu własności 4, 1, 3 i 2. □

Na ogólnie, stosując te cztery własności, możemy obliczyć symbol  $(\frac{m}{n})$  w czasie wielomianowym. Jedyne wymagane tu działania arytmetyczne to redukcje modularne oraz rozkład potęgi dwójki. Zauważmy, że jeśli liczba całkowita jest reprezentowana w zapisie binarnym, rozkład potęgi dwójki sprawdza się do ustalenia liczby zer na końcu. Tak więc o złożoności tego algorytmu decyduje liczba niezbędnych redukcji modularnych. Nietrudno udowodnić, że w przebiegu algorytmu wykonuje się co najwyżej  $O(\log n)$  takich redukcji, a każdą z nich można zrealizować w czasie  $O((\log n)^2)$ . Wynika stąd, że złożoność algorytmu wynosi  $O((\log n)^3)$ , a więc czas jego wykonania jest wielomianozależny od  $\log n$ . (W rzeczywistości, jak wykazuje bardziej szczegółowa analiza, algorytm ma złożoność rzędu  $O((\log n)^2)$ ).

Załóżmy, że wygenerowaliśmy liczbę losową  $n$  i chcemy za pomocą algorytmu Solovaya–Strassena sprawdzić, czy jest ona pierwsza. Jaką ufność możemy pokładać w odpowiedzi pozytywnej, jeśli wykonamy algorytm  $m$  razy? Chciałoby się w tym miejscu stwierdzić, że prawdopodobieństwo tego, że liczba  $n$  jest pierwsza jest wtedy równe  $1 - 2^{-m}$  – i taki wniosek jest często formułowany zarówno w podręcznikach, jak i fachowych artykułach, ale nie wynika on z posiadanych danych.

Używając prawdopodobieństw, należy zachowywać szczególną ostrożność. Określmy teraz dwie zmienne losowe. Niech  $a$  oznacza zdarzenie

„losowa nieparzysta liczba całkowita jest złożona”,

natomiast  $b$  niech oznacza zdarzenie

„algorytm odpowiada » $n$  jest liczbą pierwszą«  $m$  razy z rzędu”.

Oczywiście  $P(b|a) \leq 2^{-m}$ . Jednakże prawdopodobieństwem, które nas interesuje jest  $P(a|b)$ , na ogół różne od  $P(b|a)$ .

Do obliczenia  $P(a|b)$  możemy wykorzystać twierdzenie Bayesa (twierdzenie 2.1). Musimy w tym celu znać  $P(a)$ . Założymy, że  $N \leq n \leq 2N$ . Zgodnie

z twierdzeniem o liczbach pierwszych liczba (nieparzystych) liczb pierwszych zawartych między  $N$  i  $2N$  jest w przybliżeniu równa

$$\frac{2N}{\ln 2N} - \frac{N}{\ln N} \approx \frac{N}{\ln N} \approx \frac{n}{\ln n}.$$

Biorąc pod uwagę, że między  $N$  i  $2N$  jest  $N/2 \approx n/2$  nieparzystych liczb całkowitych, użyjemy oszacowania

$$P(\mathbf{a}) \approx 1 - \frac{2}{\ln n}.$$

Wtedy obliczenia będą wyglądały następująco:

$$\begin{aligned} P(\mathbf{a}|\mathbf{b}) &= \frac{P(\mathbf{b}|\mathbf{a})P(\mathbf{a})}{P(\mathbf{b})} \\ &= \frac{P(\mathbf{b}|\mathbf{a})P(\mathbf{a})}{P(\mathbf{b}|\mathbf{a})P(\mathbf{a}) + P(\mathbf{b}|\bar{\mathbf{a}})P(\bar{\mathbf{a}})} \\ &= \frac{P(\mathbf{b}|\mathbf{a}) \left(1 - \frac{2}{\ln n}\right)}{P(\mathbf{b}|\mathbf{a}) \left(1 - \frac{2}{\ln n}\right) + \frac{2}{\ln n}} \\ &= \frac{P(\mathbf{b}|\mathbf{a})(\ln n - 2)}{P(\mathbf{b}|\mathbf{a})(\ln n - 2) + 2} \\ &\leqslant \frac{2^{-m}(\ln n - 2)}{2^{-m}(\ln n - 2) + 2} \\ &= \frac{\ln n - 2}{\ln n - 2 + 2^{m+1}}. \end{aligned}$$

Symbol  $\bar{\mathbf{a}}$  oznacza tu zdarzenie

„losowa nieparzysta liczba całkowita jest pierwsza”.

Ciekawe jest porównanie dwóch funkcji zmiennej  $m$ :  $(\ln n - 2)/(\ln n - 2 + 2^{m+1})$  oraz  $2^{-m}$ . Przyjmijmy, że  $n \approx 2^{256} \approx e^{177}$ , gdyż taka jest wielkość liczb pierwszych, których poszukujemy dla RSA. Wówczas pierwsza z tych funkcji jest w przybliżeniu równa  $175/(175 + 2^{m+1})$ . W tablicy na rysunku 4.8 przedstawiamy wartości tej funkcji dla niektórych argumentów  $m$ .

Chociaż wartość  $175/(175 + 2^{m+1})$  dąży do 0 wykładniczo, lecz nie tak szybko jak  $2^{-m}$ . W praktyce jednak wystarczy przyjąć  $m = 50$  lub  $m = 100$ , by uzyskać bardzo małe prawdopodobieństwo błędu.

Kończymy ten podrozdział jeszcze jednym algorytmem Monte Carlo sprawdzającym czy liczba jest złożona, zwanym algorytmem Millera–Rabina (znanym także jako „mocny test liczb pseudopierwszych”). Przedstawiamy go na rysunku 4.9. Łatwo zauważyc, iż algorytm ten działa w czasie wielomianowym; z elementarnej analizy wynika, że ma on złożoność  $O((\log n)^3)$ , taką samą jak test Solovaya–Strassena. W rzeczywistości algorytm Millera–Rabina daje w praktyce lepsze wyniki niż algorytm Solovaya–Strassena.

$m$	$2^{-m}$	$\frac{175}{175 + 2^{m+1}}$
1	0,500	0,978
2	0,250	0,956
5	$0,312 \cdot 10^{-1}$	0,732
10	$0,977 \cdot 10^{-3}$	$0,787 \cdot 10^{-1}$
20	$0,954 \cdot 10^{-6}$	$0,834 \cdot 10^{-4}$
30	$0,931 \cdot 10^{-9}$	$0,815 \cdot 10^{-7}$
50	$0,888 \cdot 10^{-15}$	$0,777 \cdot 10^{-13}$
100	$0,789 \cdot 10^{-30}$	$0,690 \cdot 10^{-28}$

RYSUNEK 4.8. Prawdopodobieństwa błędu w teście Solovaya-Strasena

- (1) napisz  $n - 1 = 2^k m$ , gdzie  $m$  jest liczbą nieparzystą
- (2) wybierz losową liczbę całkowitą  $a$ ,  $1 \leq a \leq n - 1$
- (3) oblicz  $b = a^m \pmod{n}$
- (4) **if**  $b \equiv 1 \pmod{n}$  **then**  
    odpowiedz „ $n$  jest pierwsza” **QUIT**
- (5) **for**  $i = 0$  **to**  $k - 1$  **do**
- (6)     **if**  $b \equiv -1 \pmod{n}$  **then**  
        odpowiedz „ $n$  jest pierwsza” **QUIT**  
    **else**  
         $b = b^2 \pmod{n}$
- (7) odpowiedz „ $n$  jest złożona”

RYSUNEK 4.9. Test liczb pierwszych Millera-Rabina dla nieparzystej liczby całkowitej  $n$

Wykażemy teraz, że omawiany algorytm nie może udzielić odpowiedzi „ $n$  jest złożona”, gdy liczba  $n$  jest pierwsza. Inaczej mówiąc, algorytm Millera-Rabina jest pozytywnie nastawiony.

#### TWIERDZENIE 4.10

Algorytm Millera-Rabina badania złożoności liczb jest pozytywnie nastawionym algorytmem Monte Carlo.

**DOWÓD** Wykażemy tezę, doprowadzając do sprzeczności założenie, że algorytm odpowiada „ $n$  jest złożona” dla pewnej liczby pierwszej  $n$ . Taka odpowiedź następuje wtedy, gdy  $a^m \not\equiv 1 \pmod{n}$ . Rozpatrzmy teraz ciąg wartości  $b$  testowanych w algorytmie. W każdej iteracji pętli **for** liczba  $b$  jest podnoszona do kwadratu, sprawdzamy zatem wartości  $a^m, a^{2m}, \dots, a^{2^{k-1}m}$ . Ponieważ algorytm odpowiada „ $n$  jest liczbą złożoną”, wnioskujemy, że

$$a^{2^i m} \not\equiv -1 \pmod{n}$$

dla  $0 \leq i \leq k - 1$ .

Korzystając teraz z założenia, że  $n$  jest liczbą pierwszą, i stosując twierdzenie Fermata (wniosek 4.6), mamy

$$a^{2^k m} \equiv 1 \pmod{n},$$

gdyż  $n - 1 = 2^k m$ . Wówczas  $a^{2^{k-1}m}$  jest pierwiastkiem kwadratowym z 1 modulo  $n$ , czyli  $\pm 1 \pmod{n}$ . Wynika to z następujących obserwacji:  $x$  jest pierwiastkiem kwadratowym z 1 modulo  $n$  wtedy i tylko wtedy, gdy

$$n \mid (x - 1)(x + 1).$$

Ponieważ liczba  $n$  jest pierwsza, zatem albo  $n|(x - 1)$  (czyli  $x \equiv 1 \pmod{n}$ ), albo  $n|(x + 1)$  (czyli  $x \equiv -1 \pmod{n}$ ).

Wiemy, że

$$a^{2^{k-1}m} \not\equiv -1 \pmod{n},$$

musi zatem być spełniony warunek

$$a^{2^{k-1}m} \equiv 1 \pmod{n}.$$

Wtedy  $a^{2^{k-2}m}$  musi być pierwiastkiem kwadratowym z 1. Rozumując podobnie,

$$a^{2^{k-2}m} \equiv 1 \pmod{n}.$$

Powtarzając ten wywód, dochodzimy w końcu do warunku

$$a^m \equiv 1 \pmod{n},$$

co jest sprzecznością, jako że w tym przypadku algorytm odpowiedziałby „ $n$  jest pierwsza”. ■

Pozostaje zbadać prawdopodobieństwo błędu algorytmu Millera–Rabina. Można wykazać, czego nie będziemy tu robić, że nie przekracza ono  $1/4$ .

## 4.6. Ataki na RSA

Pytanie, którym zajmiemy się w tym podrozdziale, brzmi: czy można zaatakować RSA inaczej niż przez rozkład liczby  $n$  na czynniki? Zauważmy od razu, że kryptoanalitykowi wystarczy znajomość liczby  $\phi(n)$ . Jeśli znane są liczby  $n$  i  $\phi(n)$ , a  $n$  jest iloczynem dwóch liczb pierwszych  $p, q$ , to  $n$  można łatwo rozłożyć na czynniki pierwsze, rozwiązując dwa równania:

$$n = pq$$

$$\phi(n) = (p - 1)(q - 1)$$

z dwiema niewiadomymi  $p$  i  $q$ . Podstawiając  $q = n/p$  do drugiego równania, otrzymamy równanie kwadratowe zmiennej  $p$ :

$$p^2 - (n - \phi(n) + 1)p + n = 0.$$

Pierwiastkami tego równania będą liczby  $p$  i  $q$ , czynniki liczby  $n$ . Tak więc znajomość liczby  $\phi(n)$  pozwoli kryptoanalitykowi rozłożyć na czynniki  $n$  i tym samym złamać system. Innymi słowy, obliczenie  $\phi(n)$  nie jest łatwiejsze niż rozkład na czynniki liczby  $n$ .

Oto przykład.

### Przykład 4.9

Przypuśćmy, że kryptoanalityk poznał liczby  $n = 84773093$  oraz  $\phi(n) = 84754668$ . Ta informacja pozwala mu ułożyć następujące równanie kwadratowe:

$$p^2 - 18426p + 84773093 = 0.$$

Można je rozwiązać za pomocą standardowego wzoru; otrzymujemy dwa pierwiastki: 9539 i 8887. To właśnie dwa czynniki liczby  $n$ .  $\square$

#### 4.6.1. Wykładnik deszyfrowania

Udowodnimy teraz bardzo ciekawy wynik: każdy algorytm obliczający wykładnik deszyfrowania  $a$  może być użyty jako podprogram (albo *wyroczeń*) w probabilistycznym algorytmie rozkładającym  $n$  na czynniki pierwsze. Wynika stąd, że obliczenie  $a$  jest co najmniej równie trudne jak rozkład  $n$ . Nie wyklucza to jednak możliwości złamania systemu kryptograficznego bez obliczania  $a$ .

Zauważmy, że wspomniany wyżej wynik ma nie tylko teoretyczne znaczenie. Oznacza on, iż ujawnienie liczby  $a$  naraża na niebezpieczeństwo także wartość  $n$ . Jeśli tak się zdarzy, Bolek musi nie tylko zmienić wykładnik deszyfrowania – musi również wybrać nowy moduł  $n$ .

Algorytm, który za chwilę opiszemy, jest algorytmem probabilistycznym typu Las Vegas. Oto definicja.

#### DEFINICJA 4.5

Niech  $\epsilon$  będzie liczbą rzeczywistą spełniającą warunek  $0 \leq \epsilon < 1$ . *Algorytmem Las Vegas* nazywamy taki algorytm probabilistyczny, który przy rozpatrywaniu dowolnego przypadku może z pewnym prawdopodobieństwem  $\epsilon$  nie udzielić żadnej odpowiedzi (tzn. może zakończyć działanie komunikatem „brak odpowiedzi”), ale każda udzielona odpowiedź jest poprawna.

**UWAGA** Algorytm Las Vegas może nie dać rozstrzygnięcia, ale gdy je daje, jest ono poprawne, natomiast algorytm Monte Carlo zawsze daje odpowiedź, lecz nie zawsze poprawną.  $\blacksquare$

Jeśli do rozwiązyania problemu chcemy zastosować algorytm Las Vegas, to po prostu uruchamiamy go wielokrotnie, dopóki nie otrzymamy odpowiedzi. Prawdopodobieństwo braku rozstrzygnięcia po  $m$  kolejnych przebiegach jest równe  $\epsilon^m$ . Średnia (czyli oczekiwana) liczba przebiegów potrzebnych do otrzymania odpowiedzi wynosi  $1/(1 - \epsilon)$  (patrz ćwiczenia).

Załóżmy, że  $\mathbf{A}$  jest hipotetycznym algorytmem obliczającym wartość wykładownika deszyfrowania  $a$  na podstawie danych  $b$  i  $n$ . Opiszemy algorytm Las Vegas używający algorytmu  $\mathbf{A}$  jako wyroczni. Algorytm ten rozkłada liczbę  $n$  na czynniki pierwsze z prawdopodobieństwem co najmniej  $1/2$ . Jeśli więc puścimy go  $m$  razy, liczba  $n$  zostanie rozłożona z prawdopodobieństwem  $1 - 1/2^m$ .

Algorytm opiera się na pewnych faktach dotyczących pierwiastków kwadratowych z 1 modulo  $n$ , gdzie  $n = pq$  jest iloczynem dwóch różnych nieparzystych liczb pierwszych. Przypomnijmy, że kongruencja  $x^2 \equiv 1 \pmod p$  ma dwa rozwiązania modulo  $p$ , mianowicie  $x = \pm 1 \pmod p$ . Podobnie kongruencja  $x^2 \equiv 1 \pmod q$  ma dwa rozwiązania, mianowicie  $x = \pm 1 \pmod q$ .

Z tego, że  $x^2 \equiv 1 \pmod n$  wtedy i tylko wtedy, gdy  $x^2 \equiv 1 \pmod p$  i  $x^2 \equiv 1 \pmod q$ , wynika, że  $x^2 \equiv 1 \pmod n$  wtedy i tylko wtedy, gdy  $x \equiv \pm 1 \pmod p$  i  $x \equiv \pm 1 \pmod q$ . Istnieją zatem cztery pierwiastki kwadratowe z 1 modulo  $n$  i można je znaleźć za pomocą chińskiego twierdzenia o resztach. Dwa z tych rozwiązań to  $x = \pm 1 \pmod n$ ; są to tak zwane *trywialne* pierwiastki kwadratowe z 1 modulo  $p$ . Pozostałe dwa pierwiastki nazywane są *nietrywialnymi* i są elementami wzajemnie przeciwnymi modulo  $n$ .

Popatrzmy na przykład.

#### Przykład 4.10

Niech  $n = 403 = 13 \cdot 31$ . Pierwiastkami kwadratowymi z 1 modulo 403 są liczby 1, 92, 311 oraz 402. Pierwiastek 92 otrzymujemy z układu  $x \equiv 1 \pmod{13}$ ,  $x \equiv -1 \pmod{31}$  za pomocą chińskiego twierdzenia o resztach. Mając ten nietrywialny pierwiastek, wiemy, że drugim takim pierwiastkiem musi być liczba  $403 - 92 = 311$ . Jest ona rozwiązaniem układu  $x \equiv -1 \pmod{13}$  i  $x \equiv 1 \pmod{31}$ .  $\square$

Załóżmy, że  $x$  jest nietrywialnym pierwiastkiem kwadratowym z 1 modulo  $n$ . Wtedy

$$n \mid (x - 1)(x + 1),$$

ale  $n$  nie jest podzielnikiem żadnego z czynników po prawej. Stąd wynika, że  $\text{NWD}(x + 1, n) = p$  lub  $q$  (i analogicznie,  $\text{NWD}(x - 1, n) = p$  lub  $q$ ). Oczywiście największy wspólny dzielnik dwóch liczb możemy obliczyć za pomocą algorytmu Euklidesa, nie znając rozkładu liczby na czynniki  $n$ . Okazuje się zatem, że znajomość nietrywialnego pierwiastka kwadratowego z 1 modulo  $n$  pozwala obliczyć rozkład liczby  $n$  w zaledwie wielomianowym czasie obliczeń. Ten ważny fakt jest podstawą wielu wyników w kryptografii.

W przykładzie 4.10 mamy:  $\text{NWD}(93, 403) = 31$  oraz  $\text{NWD}(312, 403) = 13$ .

```

(1) wybierz losowo  $w$ , takie że  $1 \leq w \leq n - 1$ 
(2) oblicz  $x = \text{NWD}(w, n)$ 
(3) if  $1 < x < n$  then quit (sukces:  $x = p$  lub  $x = q$ )
(4) oblicz  $a = \mathbf{A}(b)$ 
(5) napisz  $ab - 1 = 2^s r$ ,  $r$  nieparzysta
(6) oblicz  $v = w^r \bmod n$ 
(7) if  $v \equiv 1 \pmod{n}$  then quit (porażka)
(8) while  $v \not\equiv 1 \pmod{n}$  do
(9)    $v_0 = v$ 
(10)   $v = v^2 \bmod n$ 
(11) if  $v_0 \equiv -1 \pmod{n}$  then
        quit (porażka)
    else
        oblicz  $x = \text{NWD}(v_0 + 1, n)$  (sukces:  $x = p$  lub  $x = q$ )

```

**RYSUNEK 4.10.** Algorytm rozkładu na czynniki na podstawie wykładnika szyfrowania  $a$

Na rysunku 4.10 przedstawiamy algorytm, który, korzystając z hipotetycznego algorytmu  $\mathbf{A}$  jako podprogramu, poszukuje nietrywialnego pierwiastka kwadratowego z 1 modulo  $n$ , by użyć go do próby rozkładu  $n$  na czynniki. (Przypomnijmy, że  $\mathbf{A}$  oblicza wykładnik deszyfrowania  $a$  odpowiadający wykładnikowi szyfrowania  $b$ ). Zobaczmy najpierw przykład ilustrujący zastosowanie tego algorytmu.

#### Przykład 4.11

Niech  $n = 89855713$ ,  $b = 34986517$  i  $a = 82330933$  oraz niech wartością losową będzie  $w = 5$ . Wówczas

$$ab - 1 = 2^3 \cdot 360059073378795.$$

W kroku 6 otrzymujemy  $v = 85877701$ , a w kroku 10 mamy  $v = 1$ . W kroku 12 obliczamy:

$$\text{NWD}(85877702, n) = 9103.$$

To jeden z czynników  $n$ ; drugim jest  $n/9103 = 9871$ . □

Przystąpmy teraz do analizy algorytmu. Po pierwsze, zauważmy, że jeśli uda nam się szczęśliwie wybrać liczbę  $w$ , tak by była wielokrotnością  $p$  lub  $q$ , to potrafimy natychmiast rozłożyć  $n$  na czynniki, co okaże się już w kroku 2. Gdy  $w$  jest względnie pierwsza z  $n$ , podnosząc kolejno do kwadratu obliczamy  $w^r, w^{2r}, w^{4r}, \dots$  aż dla pewnego  $t$  spełniony będzie warunek

$$w^{2^t r} \equiv 1 \pmod{n}.$$

Z kolei z warunku

$$ab - 1 = 2^s r \equiv 0 \pmod{\phi(n)}$$

wnioskujemy, że  $w^{2^s r} \equiv 1 \pmod{n}$ . Tak więc pętla **while** zakończy się po co najwyżej  $s$  iteracjach. Na końcu tej pętli będziemy mieli wartość  $v_0$ , taką że  $v_0^2 \equiv 1 \pmod{n}$ , ale  $v_0 \not\equiv 1 \pmod{n}$ . Jeśli  $v_0 \equiv -1 \pmod{n}$ , to algorytm nie daje wyniku; w przeciwnym przypadku  $v_0$  jest nietrywialnym pierwiastkiem kwadratowym z 1 modulo  $n$ , co prowadzi do rozkładu  $n$  na czynniki (krok 12).

Główne zadanie, które nas teraz czeka, to wykazanie, że algorytm daje wynik z prawdopodobieństwem co najmniej  $1/2$ . Brak rozkładu liczby  $n$  na czynniki może wynikać z dwóch powodów:

$$(1) \quad w^r \equiv 1 \pmod{n} \text{ (krok 7)}$$

$$(2) \quad w^{2^t r} \equiv -1 \pmod{n} \text{ dla pewnego } t, 0 \leq t \leq s-1 \text{ (krok 11).}$$

Mamy do rozważenia  $s+1$  kongruencji. Jeśli losowo wybrana liczba  $w$  stanowi rozwiązanie co najmniej jednej z nich, to „wybór” okazuje się zły i algorytm zawodzi. Policzymy zatem, ile rozwiązań ma każda z kongruencji.

Na początek zajmijmy się kongruencją  $w^r \equiv 1 \pmod{n}$ . W celu jej przeanalizowania rozpatrzmy najpierw rozwiązania modulo  $p$  i modulo  $q$  oddzielnie, a następnie połączymy je za pomocą chińskiego twierdzenia o resztach. Zauważmy, że  $x \equiv 1 \pmod{n}$  wtedy i tylko wtedy, gdy  $x \equiv 1 \pmod{p}$  oraz  $x \equiv 1 \pmod{q}$ .

Zaczniemy więc od kongruencji  $x \equiv 1 \pmod{p}$ . Ponieważ liczba  $p$  jest pierwsza, zatem na mocy twierdzenia 4.7  $\mathbb{Z}_p^*$  jest grupą cykliczną. Niech  $g$  będzie elementem pierwotnym modulo  $p$ . Dla pewnej jednoznacznie określonej liczby całkowitej  $u$ , takiej że  $0 \leq u \leq p-2$ , mamy  $w = g^u$ . Wówczas

$$w^r \equiv 1 \pmod{p}$$

$$g^{ur} \equiv 1 \pmod{p}$$

$$(p-1) \mid ur.$$

Niech

$$p-1 = 2^i p_1$$

dla liczby nieparzystej  $p_1$  oraz

$$q-1 = 2^j q_1$$

dla pewnej liczby nieparzystej  $q_1$ . Z warunku

$$\phi(n) = (p-1)(q-1) \mid (ab-1) = 2^s r$$

mamy

$$2^{i+j} p_1 q_1 \mid 2^s r,$$

a stąd

$$i + j \leq s$$

oraz

$$p_1 q_1 \mid r.$$

Dalej, warunek  $(p - 1) \mid ur$  przybiera postać  $2^i p_1 \mid ur$ . Ponieważ  $p_1 \mid r$  i  $r$  jest liczbą nieparzystą, warunkiem koniecznym i wystarczającym do jego spełnienia jest, by  $2^i \mid u$ . Stąd  $u = k2^i$ ,  $0 \leq k \leq p_1 - 1$ , a kongruencja  $u^r \equiv 1 \pmod{p}$  ma  $p_1$  rozwiązań.

Rozumując podobnie, wnosimy, że kongruencja  $w^r \equiv 1 \pmod{q}$  ma dokładnie  $q_1$  rozwiązań. Łącząc każde rozwiązanie modulo  $p$  z dowolnym rozwiązaniem modulo  $q$ , otrzymujemy, stosując chińskie twierdzenie o resztach, dokładnie jedno rozwiązanie modulo  $n$ . W rezultacie liczba rozwiązań kongruencji  $w^r \equiv 1 \pmod{n}$  jest równa  $p_1 q_1$ .

Następny etap to rozważenie kongruencji  $w^{2^t r} \equiv -1 \pmod{n}$  dla ustalonej wartości  $t$  (gdzie  $0 \leq t \leq s - 1$ ). I znów, zaczynamy od kongruencji modulo  $p$  i modulo  $q$  (zauważmy, że  $w^{2^t r} \equiv -1 \pmod{n}$  wtedy i tylko wtedy, gdy  $w^{2^t r} \equiv -1 \pmod{p}$  i  $w^{2^t r} \equiv -1 \pmod{q}$ ). Zajmijmy się najpierw kongruencją  $w^{2^t r} \equiv -1 \pmod{p}$ . Jeśli, jak poprzednio, przyjmiemy  $w = g^u$ , to mamy

$$g^{u2^t r} \equiv -1 \pmod{p}.$$

Porównując to z warunkiem  $g^{(p-1)/2} \equiv -1 \pmod{p}$ , otrzymujemy

$$u2^t r \equiv \frac{p-1}{2} \pmod{p-1}$$

$$(p-1) \mid \left( u2^t r - \frac{p-1}{2} \right)$$

$$2(p-1) \mid (u2^t r - (p-1)).$$

Z kolei z równości  $p-1 = 2^i p_1$  mamy

$$2^{i+1} p_1 \mid (u2^{t+1} r - 2^i p_1).$$

Gdy wyłączymy po obu stronach wspólny czynnik  $p_1$  i wykonamy redukcję, otrzymamy

$$2^{i+1} \mid \left( \frac{u2^{t+1} r}{p_1} - 2^i \right).$$

Jeśli  $t \geq i$ , to rozwiązań nie ma, gdyż  $2^{i+1} \mid 2^{t+1}$ , ale  $2^{i+1} \nmid 2^i$ . Z drugiej strony, gdy  $t \leq i-1$ , wówczas  $u$  jest rozwiązaniem wtedy tylko wtedy, gdy  $u$  jest nieparzystą wielokrotnością liczby  $2^{i-t-1}$  (zauważmy, że  $r/p_1$  jest nieparzystą liczbą całkowitą). Tak więc w tym przypadku liczba rozwiązań jest równa

$$\frac{p-1}{2^{i-t-1}} \cdot \frac{1}{2} = 2^t p_1.$$

Podobne rozumowanie prowadzi do wniosku, że kongruencja  $w^{2^t r} \equiv -1 \pmod{q}$  nie ma rozwiązań, gdy  $t \geq j$ , i ma  $2^t q_1$  rozwiązań, gdy  $t \leq j-1$ . Stosując ponownie chińskie twierdzenie o resztach, stwierdzamy, że liczba rozwiązań kongruencji  $w^{2^t r} \equiv -1 \pmod{n}$  jest równa

$$0, \quad \text{gdy } t \geq \min\{i, j\} \\ 2^{2t} p_1 q_1, \quad \text{gdy } t \leq \min\{i, j\} - 1.$$

Zmienna  $t$  może przybierać wartości od 0 do  $s-1$ . Bez utraty ogólności możemy przyjąć, że  $i \leq j$ ; wtedy, gdy  $t \geq i$ , liczba rozwiązań jest równa 0. Ogólna liczba „złych” wyborów liczby  $w$  nie przekracza liczby

$$p_1 q_1 + p_1 q_1 (1 + 2^2 + 2^4 + \dots + 2^{2i-2}) = p_1 q_1 \left(1 + \frac{2^{2i} - 1}{3}\right) \\ = p_1 q_1 \left(\frac{2}{3} + \frac{2^{2i}}{3}\right).$$

Przypomnijmy, że  $p-1 = 2^i p_1$  oraz  $q-1 = 2^j q_1$ . Ponadto  $p_1 q_1 < n/4$ , gdyż  $j \geq i \geq 1$ . Wiemy też, że

$$2^{2i} p_1 q_1 \leq 2^{i+j} p_1 q_1 = (p-1)(q-1) < n.$$

Otrzymujemy stąd

$$p_1 q_1 \left(\frac{2}{3} + \frac{2^{2i}}{3}\right) < \frac{n}{6} + \frac{n}{3} \\ = \frac{n}{2}.$$

Ponieważ co najwyżej  $(n-1)/2$  wyborów liczby  $w$  jest „złych”, mamy co najmniej  $(n-1)/2$  wyborów „dobrych”. Stąd prawdopodobieństwo sukcesu algorytmu wynosi co najmniej  $1/2$ .

#### 4.6.2. Częściowa informacja o bitach tekstu jawnego

Następny wynik, którym się teraz zajmiemy, dotyczy częściowej informacji o tekście jawnym, która może „wycieć” podczas szyfrowania w systemie RSA. Rozpatrujemy tu dwa przykłady częściowej informacji:

- (1) Mając dane  $y = e_K(x)$ , oblicz  $\text{par}(y)$ , gdzie  $\text{par}(y)$  oznacza najmniej znaczący bit  $x$ ;
- (2) Mając dane  $y = e_K(x)$ , oblicz  $\text{pol}(y)$ , gdzie  $\text{pol}(y) = 0$ , gdy  $0 \leq x < n/2$ , oraz  $\text{pol}(y) = 1$ , gdy  $n/2 < x \leq n-1$ .

Wykażemy, że każdy algorytm, który dla danego  $y = e_K(x)$  oblicza  $\text{par}(y)$  i  $\text{pol}(y)$ , może służyć za wyrocznię przy konstrukcji algorytmu obliczającego tekst

jawnego  $x$ . To znaczy, innymi słowy, że obliczanie najbliższego bitu tekstu jawnego na podstawie znanego tekstu zaszyfrowanego jest wielomianowo równoważne ustaleniu pełnego tekstu jawnego!

Najpierw udowodnimy, że obliczanie  $\text{par}(y)$  jest wielomianowo równoważne obliczaniu  $\text{pol}(y)$ . Wynika to z następujących łatwych do wykazania równości (patrz ćwiczenia):

$$\text{pol}(y) = \text{par}(y \cdot e_K(2) \bmod n) \quad (4.1)$$

$$\text{par}(y) = \text{pol}(y \cdot e_K(2^{-1}) \bmod n) \quad (4.2)$$

oraz z prawa mnożenia  $e_K(x_1)e_K(x_2) = e_K(x_1x_2)$ .

Pokażemy, jak obliczyć  $x = d_K(y)$ , dysponując hipotetycznym algorytmem (wyrocznią) obliczającym  $\text{pol}(y)$ . Algorytm przedstawiamy na rysunku 4.11.

```

(1) oznacz  $k = \lfloor \log_2 n \rfloor$ 
(2) for  $i = 0$  to  $k$  do
(3)    $y_i = \text{pol}(y)$ 
(4)    $y = (y \cdot e_K(2)) \bmod n$ 
(5)    $lo = 0$ 
(6)    $hi = n$ 
(7)   for  $i = 0$  to  $k$  do
(8)      $mid = (hi + lo)/2$ 
(9)     if  $y_i = 1$  then
           $lo = mid$ 
        else
           $hi = mid$ 
(10)   $x = \lfloor hi \rfloor$ 
```

RYSUNEK 4.11. Deszyfrowanie tekstu zaszyfrowanego w RSA za pomocą wyroczni obliczającej  $\text{pol}(y)$

W krokach 2–4 obliczamy

$$y_i = \text{pol}(y \cdot (e_K(2))^i) = \text{pol}(e_K(x \cdot 2^i))$$

dla  $0 \leq i \leq \log_2 n$ . Zauważamy, że

$$\text{pol}(e_K(x)) = 0 \Leftrightarrow x \in \left[0, \frac{n}{2}\right)$$

$$\text{pol}(e_K(2x)) = 0 \Leftrightarrow x \in \left[0, \frac{n}{4}\right) \cup \left[\frac{n}{2}, \frac{3n}{4}\right)$$

$$\text{pol}(e_K(4x)) = 0 \Leftrightarrow x \in \left[0, \frac{n}{8}\right) \cup \left[\frac{n}{4}, \frac{3n}{8}\right) \cup \left[\frac{n}{2}, \frac{5n}{8}\right) \cup \left[\frac{3n}{4}, \frac{7n}{8}\right)$$

i tak dalej. Możemy zatem znaleźć  $x$  za pomocą przeszukiwania binarnego, co odbywa się w krokach 7–11. Oto przykład dla ilustracji.

**Przykład 4.12**

Niech  $n = 1457$ ,  $b = 779$  i niech tekstem zaszyfrowanym będzie  $y = 722$ . Obliczamy  $e_K(2) = 946$ . Założmy, że stosując wyrocznię dla funkcji *pol* otrzymujemy w kroku 3 algorytmu następujące wartości  $y_i$ :

$i$	0	1	2	3	4	5	6	7	8	9	10
$y_i$	10	1	0	1	1	1	1	1	0	0	

Wówczas przeszukiwanie binarne przebiega tak, jak to widać na rysunku 4.12. Tekstem jawnym jest więc  $x = [999, 55] = 999$ .  $\square$

$i$	$lo$	$mid$	$hi$
0	0,00	728,50	1457,00
1	728,50	1092,75	1457,00
2	728,50	910,62	1092,75
3	910,62	1001,69	1092,75
4	910,62	956,16	1001,69
5	956,16	978,92	1001,69
6	978,92	990,30	1001,69
7	990,30	996,00	1001,69
8	996,00	998,84	1001,69
9	998,84	1000,26	1001,69
10	998,84	999,55	1000,26
	998,84	999,55	999,55

RYSUNEK 4.12. Przeszukiwanie binarne podczas deszyfrowania RSA

## 4.7. System kryptograficzny Rabina

Przedstawimy w tym podrozdziale **system kryptograficzny Rabina**, który jest obliczeniowo bezpieczny na wypadek ataku opartego na wybranym tekście jawnym, jeśli modułu  $n = pq$  nie można rozłożyć na czynniki. Na rysunku 4.13 znajduje się krótka opis systemu.

Wykażemy, że funkcja szyfrowania  $e_K$  nie jest różnowartościowa, a więc deszyfrowanie nie jest jednoznaczne. Dany tekst zaszyfrowany może reprezentować cztery możliwe teksty jawne. Dokładniej, niech  $\omega$  będzie jednym z pierwiastków kwadratowych z 1 modulo  $n$ . Niech  $x \in \mathbb{Z}_n$ . Możemy wtedy przekonać się

Niech  $n$  będzie iloczynem dwóch różnych liczb pierwszych  $p$  i  $q$ ,  $p, q \equiv 3 \pmod{4}$ . Niech dalej  $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$ . Określmy

$$\mathcal{K} = \{(n, p, q, B) : 0 \leq B \leq n - 1\}.$$

Dla  $K = (n, p, q, B)$  definiujemy

$$e_K(x) = x(x + B) \bmod n$$

oraz

$$d_K(y) = \sqrt{\frac{B^2}{4} + y} - \frac{B}{2} \bmod n.$$

Wartości  $n$  oraz  $B$  są znane publicznie, natomiast  $p$  i  $q$  pozostają tajne.

#### RYSUNEK 4.13. System kryptograficzny Rabina

o prawdziwości następujących równości:

$$\begin{aligned} e_K\left(\omega\left(x + \frac{B}{2}\right) - \frac{B}{2}\right) &= \left(\omega\left(x + \frac{B}{2}\right) - \frac{B}{2}\right)\left(\omega\left(x + \frac{B}{2}\right) + \frac{B}{2}\right) \\ &= \omega^2\left(x + \frac{B}{2}\right)^2 - \left(\frac{B}{2}\right)^2 \\ &= x^2 + Bx + \frac{B^2}{4} - \frac{B^2}{4} \\ &= x^2 + Bx \\ &= e_K(x). \end{aligned}$$

(Zauważmy, że wszystkie obliczenia są wykonywane w  $\mathbb{Z}_n$ , więc dzielenie przez 2 i przez 4 jest tym samym, co mnożenie przez  $2^{-1}$  i  $4^{-1}$  modulo  $n$ , odpowiednio).

Czterem tekstom jawnym:  $x, -x - B, \omega(x + B/2) - B/2$  oraz  $-\omega(x + B/2) - B/2$ , gdzie  $\omega$  jest nietrywialnym pierwiastkiem kwadratowym z 1 modulo  $n$ , przyporządkowany jest ten sam szyfr  $e_K(x)$ . Na ogół Bolek nie jest w stanie określić, który z tych tekstów jest „poprawny”, o ile tekst jawny nie zawiera nadmiarowości wystarczającej do wyeliminowania trzech spośród czterech możliwych wartości.

Popatrzmy na problem deszyfrowania z punktu widzenia Bolka, który dostaje tekst zaszyfrowany  $y$  i chce ustalić wartość  $x$ , taką by

$$x^2 + Bx \equiv y \pmod{n}.$$

Mamy tu równanie kwadratowe zmiennej  $x$ . Podstawiając  $x_1 = x + B/2$  lub równoważnie  $x = x_1 - B/2$ , możemy wyeliminować składnik liniowy. Dochodzimy wtedy do równania

$$x_1^2 - Bx_1 + \frac{B^2}{4} + Bx_1 - \frac{B^2}{2} - y \equiv 0 \pmod{n}$$

lub

$$x_1^2 \equiv \frac{B^2}{4} + y \pmod{n}.$$

Jeśli przyjmiemy  $C = B^2/4 + y$ , to możemy ten warunek zapisać krócej:

$$x_1^2 \equiv C \pmod{n}.$$

Widać więc, że deszyfrowanie sprowadza się do obliczania pierwiastków kwadratowych modulo  $n$  lub – równoważnie – do rozwiązywania pary kongruencji:

$$x_1^2 \equiv C \pmod{p}$$

oraz

$$x_1^2 \equiv C \pmod{q}.$$

(Istnieją dwa pierwiastki kwadratowe z  $C$  modulo  $p$  oraz dwa pierwiastki kwadratowe z  $C$  modulo  $q$ . Korzystając z chińskiego twierdzenia o resztach, otrzymujemy z nich cztery pierwiastki z  $C$  modulo  $N$ ). Za pomocą kryterium Eulera możemy ustalić, czy  $C$  jest resztą kwadratową modulo  $p$  (i modulo  $q$ ), co będzie miało miejsce, jeśli szyfrowanie zostało poprawnie przeprowadzone. Kryterium Eulera nie ułatwia jednak znalezienia pierwiastków kwadratowych z  $C$ . Wynikiem będzie tylko odpowiedź „tak” lub „nie”.

Gdy  $p \equiv 3 \pmod{4}$ , do obliczenia pierwiastków kwadratowych z resztą kwadratowymi modulo  $p$  można użyć prostego wzoru. Niech  $C$  będzie resztą kwadratową. Założymy też, że  $p \equiv 3 \pmod{4}$ . Wówczas

$$\begin{aligned} (\pm C^{(p+1)/4})^2 &\equiv C^{(p+1)/2} \pmod{p} \\ &\equiv C^{(p-1)/2}C \pmod{p} \\ &\equiv C \pmod{p}. \end{aligned}$$

Skorzystajmy znów z kryterium Eulera, które stwierdza, że jeśli  $C$  jest resztą kwadratową modulo  $p$ , to  $C^{(p-1)/2} \equiv 1 \pmod{p}$ . Tak więc pierwiastkami kwadratowymi z  $C$  modulo  $p$  są liczby  $\pm C^{(p+1)/4} \pmod{p}$ . Podobnie, pierwiastkami kwadratowymi z  $C$  modulo  $q$  są  $\pm C^{(q+1)/4} \pmod{q}$ . Stąd już bez trudu otrzymujemy za pomocą chińskiego twierdzenia o resztach cztery pierwiastki kwadratowe  $x_1$  z  $C$  modulo  $n$ .

**UWAGA** Co ciekawe, nie jest znany żaden algorytm deterministyczny obliczający w czasie wielomianowym pierwiastki kwadratowe z resztą kwadratowymi modulo  $p$ , gdy  $p \equiv 1 \pmod{4}$ . Istnieje natomiast działający w czasie wielomianowym algorytm typu Las Vegas.

Po ustaleniu czterech możliwych wartości  $x_1$  obliczamy  $x$  z równania  $x = x_1 - B/2$  i otrzymujemy cztery możliwe teksty jawne. Uzyskujemy w ten sposób regułę deszyfrowania:

$$d_K(y) = \sqrt{\frac{B^2}{4} + y} - \frac{B}{2}.$$

**Przykład 4.13**

Do ilustracji procesu szyfrowania i deszyfrowania w systemie kryptograficznym Rabina użyjemy bardzo prostego przykładu. Niech  $n = 77 = 7 \cdot 11$  i  $B = 9$ . Wówczas funkcja szyfrowania ma postać

$$e_K(x) = x^2 + 9x \pmod{77},$$

a deszyfrowania

$$d_K(y) = \sqrt{1+y} - 43 \pmod{77}.$$

Przypuśćmy, że Bolek chce odczytać tekst zaszyfrowany  $y = 22$ . W tym celu musi przede wszystkim obliczyć pierwiastki kwadratowe z 23 modulo 7 i modulo 11. Obie te liczby, 7 i 11, przystają do 3 modulo 4, możemy więc użyć wzorów

$$23^{(7+1)/4} \equiv 2^2 \equiv 4 \pmod{7}$$

oraz

$$23^{(11+1)/4} \equiv 1^3 \equiv 1 \pmod{11}.$$

Stosując chińskie twierdzenie o resztach, obliczamy cztery pierwiastki kwadratowe z 23 modulo 77; są nimi liczby  $\pm 10, \pm 32 \pmod{77}$ . Ostatecznie dochodzimy do następujących czterech tekstów jawnych:

$$10 - 43 \pmod{77} = 44$$

$$67 - 43 \pmod{77} = 24$$

$$32 - 43 \pmod{77} = 66$$

$$45 - 43 \pmod{77} = 2.$$

Czytelnik może sprawdzić, że każdemu z nich odpowiada tekst zaszyfrowany 22.  $\square$

Zajmiemy się teraz bezpieczeństwem systemu Rabina. Wykażemy, że każdy hipotetyczny algorytm deszyfrowania **A** może być użyty jako wyrocznia w algorytmie typu Las Vegas, który rozkłada moduł  $n$  na czynniki z prawdopodobieństwem  $1/2$ . Taki algorytm jest przedstawiony na rysunku 4.14.

Musimy tu wyjaśnić kilka spraw. Po pierwsze, zauważmy, że

$$y = e_K \left( r - \frac{B}{2} \right),$$

zatem wartość  $x$  otrzymamy w kroku 3. Dalej, patrząc na krok 4, stwierdzamy, że  $x_1^2 \equiv r^2 \pmod{n}$ . Oznacza to, że  $x_1 \equiv \pm r \pmod{n}$  lub  $x_1 \equiv \pm \omega r \pmod{n}$ , gdzie  $\omega$  jest jednym z nietrywialnych pierwiastków kwadratowych z 1 modulo  $n$ . W drugim przypadku spełniony jest warunek

$$n \mid (x_1 - r)(x_1 + r),$$

```

(1) wybierz losowo  $r$ ,  $1 \leq r \leq n - 1$ 
(2) oblicz  $y = r^2 - B^2/4 \pmod{n}$ 
(3) wywołaj  $\mathbf{A}(y)$ , by uzyskać rozszyfrowany tekst  $x$ 
(4) oblicz  $x_1 = x + B/2$ 
(5) if  $x_1 \equiv \pm r \pmod{n}$  then
    quit (porażka)
else
     $\text{NWD}(x_1 + r, n) = p$  lub  $q$  (sukces)

```

**RYSUNEK 4.14.** Rozkład na czynniki modułu w systemie Rabina z użyciem wyroczni deszyfrowania

lecz  $n$  nie dzieli żadnego z czynników po prawej stronie. Tak więc wynikiem obliczenia  $\text{NWD}(x_1 + r, n)$  (lub  $\text{NWD}(x_1 - r, n)$ ) musi być albo  $p$ , albo  $q$ , co kończy proces rozkładu liczby  $n$  na czynniki.

Obliczmy prawdopodobieństwo sukcesu tego algorytmu, biorąc pod uwagę wszystkie  $n - 1$  możliwe wybory losowej wartości  $r$ . Dla dwóch niezerowych reszt  $r_1$  i  $r_2$ , niech

$$r_1 \sim r_2 \Leftrightarrow r_1^2 \equiv r_2^2 \pmod{n}.$$

Łatwo zauważyc, że  $r \sim r$  dla każdego  $r$ . Ponadto, jeśli  $r_1 \sim r_2$ , to  $r_2 \sim r_1$ , a warunki  $r_1 \sim r_2$  i  $r_2 \sim r_3$  implikują  $r_1 \sim r_3$ . Wynika stąd, że  $\sim$  jest relacją równoważności. Wszystkie jej klasy abstrakcji w  $\mathbb{Z}_n \setminus \{0\}$  są czteroelementowe; klasą abstrakcji elementu  $r$  jest zbiór

$$[r] = \{\pm r, \pm \omega r \pmod{n}\},$$

gdzie  $\omega$  jest nietrywialnym pierwiastkiem kwadratowym z 1 modulo  $n$ .

W algorytmie przedstawionym na rysunku 4.14 dowolne dwie wartości  $r$  należące do tej samej klasy abstrakcji dają tę samą wartość  $y$ . Rozpatrzmy teraz wartość  $x$  wskazaną przez wyrocznię  $\mathbf{A}$  po wprowadzeniu do niej wartości  $y$ . Mamy wtedy

$$[y] = \{\pm y, \pm \omega y\}.$$

Jeśli  $r = \pm y$ , algorytm kończy się niepowodzeniem, natomiast  $r = \pm \omega y$  zapewnia sukces algorytmu. Wartość  $r$  jest wybierana losowo, zatem prawdopodobieństwa wystąpienia wszystkich czterech możliwych wartości są równe. Wnioskujemy stąd, że prawdopodobieństwo sukcesu algorytmu jest równe  $1/2$ .

Ciekawe, że można udowodnić bezpieczeństwo systemu kryptograficznego Rabina na wypadek ataku z wybranym tekstem jawnym, natomiast jest on zupełnie tego bezpieczeństwa pozbawiony w sytuacji ataku z wybranym tekstem zaszyfrowanym. W rzeczywistości, algorytm opisany na rys. 4.14, użyty w dowodzie bezpieczeństwa wobec ataku z wybranym tekstem jawnym, może posłużyć także

do złamania systemu Rabina przy ataku z wybranym tekstem zaszyfrowanym! Podczas takiego ataku wyrocznię **A** zastępuje algorytm deszyfrowania stosowany przez Bolka.

## 4.8. Algorytmy faktoryzacji

Literatura na temat algorytmów faktoryzacji jest niezwykle obszerna, a dokładne przedstawienie tego tematu zajęłoby więcej stron niż liczy ta książka. Poprzestańmy zatem jedynie na krótkim przeglądzie wraz z omówieniem najlepszych istniejących algorytmów rozkładu i ich zastosowania w praktyce. Dla dużych liczb najbardziej efektywne są: sito kwadratowe, algorytm krzywych eliptycznych oraz sito ciała liczbowego. Powszechnie znane są też algorytmy, które można uznać za poprzedników tych trzech, czyli metoda  $\rho$  i algorytm  $p - 1$  Pollarda, algorytm  $p + 1$  Williamsa, algorytm ułamków łańcuchowych oraz, rzecz jasna, algorytm dzielenia próbnego.

W tym podrozdziale przyjmujemy, że faktoryzacja dotyczy liczby nieparzystej  $n$ . *Dzielenie próbne* polega na dzieleniu  $n$  przez każdą liczbę nieparzystą mniejszą od  $\lfloor \sqrt{n} \rfloor$ . Dla liczb  $n$ , takich że, powiedzmy,  $n < 10^{12}$ , jest to sposób całkiem rozsądny. Jednak dla większych  $n$  potrzebne są na ogół metody bardziej wyrafinowane.

### 4.8.1. Metoda $p - 1$

Opiszemy tu pochodzący z 1974 roku algorytm  $p - 1$  Pollarda, który może posłużyć za przykład prostego algorytmu, dającego się niekiedy stosować do dużych liczb. Algorytm, przedstawiony na rysunku 4.15, ma dwie dane wejściowe: liczbę  $n$  (nieparzystą), podlegającą faktoryzacji, oraz „ograniczenie”  $B$ . Oto jak działa algorytm  $p - 1$ .

Dane wejściowe:  $n$  i  $B$

- (1)  $a = 2$
- (2) **for**  $j = 2$  **do B do**  
 $a = a^j \bmod n$
- (3)  $d = \text{NWD}(a - 1, n)$
- (4) **if**  $1 < d < n$  **then**  
 $d$  jest dzielnikiem  $n$  (sukces)
- else**  
nie znaleziono dzielnika  $n$  (porażka)

RYSUNEK 4.15. Algorytm  $p - 1$  rozkładu na czynniki pierwsze

Załóżmy, że  $p$  jest dzielnikiem pierwszym liczby  $n$  oraz  $q \leq B$  dla każdej potęgi liczby pierwszej  $q$ , takiej że  $q|(p-1)$ . Musi być wówczas spełniony warunek

$$(p-1) \mid B!$$

Na końcu pętli **for** (krok 2) mamy

$$a \equiv 2^{B!} \pmod{n},$$

zatem

$$a \equiv 2^{B!} \pmod{p},$$

ponieważ  $p|n$ . Dalej,

$$2^{p-1} \equiv 1 \pmod{p}$$

na mocy twierdzenia Fermata. Z tego, że  $(p-1)|B!$ , wnosimy, iż

$$a \equiv 1 \pmod{p}$$

(w kroku 3). Tak więc w kroku 4

$$p \mid (a - 1)$$

oraz

$$p \mid n,$$

a stąd

$$p \mid d = \text{NWD}(a - 1, n).$$

Liczba  $d$  będzie nietrywialnym dzielnikiem  $n$  (o ile w kroku 3 nie pojawiło się  $a = 1$ ). Po znalezieniu nietrywialnego czynnika  $d$  możemy przejść do prób faktoryzacji  $d$  oraz  $n/d$ , jeśli są to liczby złożone.

Oto przykład dla ilustracji.

#### Przykład 4.14

Niech  $n = 15770708441$ . Jeśli zastosujemy algorytm  $p-1$  dla  $B = 180$ , w kroku 3 otrzymamy  $a = 11620221425$ , a obliczoną wartością  $d$  będzie 135979. Pełny rozkład  $n$  na czynniki pierwsze jest następujący:

$$15770708441 = 135979 \cdot 115979.$$

W tym przypadku faktoryzacja zakończyła się powodzeniem, ponieważ liczba 135978 ma tylko „małe” czynniki pierwsze:

$$135978 = 2 \cdot 3 \cdot 131 \cdot 173.$$

Jeśli więc weźmiemy  $B \geq 173$ , żądany warunek  $135978|B!$  będzie spełniony. □

W trakcie realizacji algorytmu wykonuje się  $B - 1$  modularnych potęgowania, każde wymagające co najwyżej  $2 \log_2 B$  modularnych mnożeń (jeśli będziemy podnosili do kwadratu i wymnażali). Obliczanie największego wspólnego dzielnika można wykonać w czasie  $O((\log n)^3)$ , używając algorytmu Euklidesa. W konsekwencji algorytm ma złożoność  $O(B \log B (\log n)^2 + (\log n)^3)$ . Jeśli liczba całkowita  $B$  jest wielkością rzędu  $O((\log n)^i)$  dla pewnej ustalonej liczby całkowitej  $i$ , to algorytm będzie rzeczywiście działał w czasie wielomianowym. Jednakże przy takim wyborze liczby  $B$  prawdopodobieństwo sukcesu będzie znikome. Z drugiej strony, jeśli drastycznie zwiększymy wielkość  $B$ , powiedzmy do  $\sqrt{n}$ , algorytm zakończy się sukcesem, ale jego szybkość nie będzie większa niż szybkość dzielenia próbnego.

Tak więc wada tej metody polega na tym, że wymaga ona, by liczba  $n$  miała czynnik pierwszy  $p$ , którego poprzednik  $p - 1$  rozkłada się tylko na „małe” czynniki pierwsze. Łatwo byłoby zbudować moduł RSA  $n = pq$ , który oparłby się skutecznie faktoryzacji przeprowadzonej tym sposobem. Wystarczyłoby znaleźć dwie duże liczby pierwsze  $p_1$  i  $q_1$ , takie że liczby  $p = 2p_1 + 1$  i  $q = 2q_1 + 1$  są również pierwsze (za pomocą jednego z algorytmów typu Monte Carlo sprawdzających pierwszość liczby, omówionych w podrozdziale 4.5). Wówczas modułu RSA  $n = pq$  nie dałoby się rozłożyć na czynniki metodą  $p - 1$ .

Silniejszy od niej algorytm krzywych eliptycznych, opracowany przez Lenstre w połowie lat osiemdziesiątych ubiegłego wieku, jest w istocie uogólnieniem metody  $p - 1$ . Nie będziemy tu w ogóle omawiać jego teorii, wspomnijmy tylko, że jego sukces wiąże się ze znalezieniem liczby „bliskiej”  $p$ , która ma tylko „małe” czynniki pierwsze, co zdarza się częściej. O ile metoda  $p - 1$  opiera się na właściwościach grupy  $\mathbb{Z}_p$ , o tyle metoda krzywych eliptycznych odwołuje się do grup określonych na krzywych eliptycznych modulo  $p$ .

#### 4.8.2. Algorytm Dixona i sito kwadratowe

Algorytm Dixona wywodzi się z bardzo prostego pomysłu, z którym zetknęliśmy się już w związku z systemem kryptograficznym Rabina. Opiera się on na tym, że jeśli możemy znaleźć liczbę  $x$ , taką że  $x \not\equiv \pm y \pmod{n}$  oraz  $x^2 \equiv y^2 \pmod{n}$ , to NWD( $x - y, n$ ) jest nietrywialnym czynnikiem  $n$ .

Metoda zakłada istnienie *bazy rozkładu*, czyli zbioru  $\mathcal{B}$  „małych” liczb pierwszych. Najpierw tworzymy liczby całkowite  $x$  o tej własności, że wszystkie dzielniki pierwsze liczby  $x^2 \pmod{n}$  znajdują się w zbiorze  $\mathcal{B}$  (sposób ich otrzymywania omówimy nieco później). Następnie należy wziąć iloczyn kilku takich liczb, tak aby każda liczba pierwsza z bazy rozkładu wystąpiła w nim parzystą liczbę razy. Otrzymujemy wówczas pożądaną kongruencję typu  $x^2 \equiv y^2 \pmod{n}$ , co (mamy nadzieję) doprowadzi nas do rozkładu na czynniki pierwsze liczby  $n$ .

Zobaczmy to na starannie dobranym przykładzie.

*Przykład 4.15*

Niech  $n = 15770708441$  (ta sama liczba, którą użyliśmy w przykładzie 4.14) i niech  $\mathcal{B} = \{2, 3, 5, 7, 11, 13\}$ . Rozpatrzmy trzy kongruencje:

$$8340934156^2 \equiv 3 \cdot 7 \pmod{n}$$

$$12044942944^2 \equiv 2 \cdot 7 \cdot 13 \pmod{n}$$

$$2773700011^2 \equiv 2 \cdot 3 \cdot 13 \pmod{n}.$$

Biorąc iloczyn tych trzech kongruencji, otrzymujemy

$$(8340934156 \cdot 12044942944 \cdot 2773700011)^2 \equiv (2 \cdot 3 \cdot 7 \cdot 13)^2 \pmod{n}.$$

Wymnażając wyrażenia w nawiasach modulo  $n$ , mamy

$$9503435785^2 \equiv 546^2 \pmod{n}.$$

Teraz obliczamy największy wspólny dzielnik:

$$\text{NWD}(9503435785 - 546, 15770708441) = 115759,$$

znajdując w ten sposób czynnik 115759 liczby  $n$ . □

Załóżmy, że bazą rozkładu jest zbiór  $\mathcal{B} = \{p_1, \dots, p_B\}$ . Niech  $C$  będzie liczbą nieznacznie większą od  $B$  (np.  $C = B + 10$ ) i przypuśćmy, że otrzymaliśmy  $C$  kongruencji:

$$x_j^2 \equiv p_1^{\alpha_{1j}} \cdot p_2^{\alpha_{2j}} \cdots p_B^{\alpha_{Bj}} \pmod{n}$$

dla  $1 \leq j \leq C$ . Dla każdego  $j$  rozpatrzmy wektor

$$a_j = (\alpha_{1j} \bmod 2, \dots, \alpha_{Bj} \bmod 2) \in (\mathbb{Z}_2)^B.$$

Jeśli można spośród tych wektorów wybrać takie  $a_j$ , których sumą modulo 2 jest wektor zerowy  $(0, \dots, 0)$ , to w iloczynie odpowiednich elementów  $x_j$  każdy czynnik ze zbioru  $\mathcal{B}$  wystąpi parzystą liczbę razy.

Wróćmy do przykładu 4.15, w którym te zależności mają miejsce, choć w tym przypadku  $C < B$ .

*Przykład 4.15 cd.*

Mamy trzy wektory

$$a_1 = (0, 1, 0, 1, 0, 0)$$

$$a_2 = (1, 0, 0, 1, 0, 1)$$

$$a_3 = (1, 1, 0, 0, 0, 1).$$

Łatwo spostrzec, że

$$a_1 + a_2 + a_3 = (0, 0, 0, 0, 0, 0) \bmod 2.$$

Otrzymujemy stąd znaną już nam kongruencję, prowadzącą do rozkładu liczby  $n$  na czynniki.  $\square$

Zauważmy, że znalezienie podzbioru złożonego z  $C$  wektorów  $a_1, \dots, a_C$ , których suma modulo 2 jest wektorem zerowym, jest w istocie stwierdzeniem zależności liniowej tych wektorów nad  $\mathbb{Z}_2$ . Gdy  $C > B$ , wektory muszą być zależne, co można łatwo wykazać za pomocą metody eliminacji Gaussa. Jeśli bierzemy  $C > B + 1$ , to dlatego, że nie mamy żadnej pewności, że jakakolwiek dana kongruencja doprowadzi do faktoryzacji liczby  $n$ . Średnio w 50% przypadków okaże się, że  $x \equiv \pm y \pmod{n}$ . Jednakże gdy  $C > B + 1$ , możemy otrzymać takich kongruencji więcej (odpowiadających różnym układom wektorów zależnych). Można oczekwać, że co najmniej jedna z nich umożliwi faktoryzację  $n$ .

Pozostaje jeszcze wyjaśnić, jak otrzymać liczby całkowite  $x_j$ , takie że wszystkie czynniki pierwsze liczb  $x_j^2$  mod  $n$  znajdują się w zbiorze  $\mathcal{B}$ . Można to zrobić różnymi metodami. Jedną z częściej stosowanych jest metoda sita kwadratowego Pomerance'a wykorzystująca liczby całkowite postaci  $x_j = j + \lfloor \sqrt{n} \rfloor$ ,  $j = 1, 2, \dots$ . Nazwa „sito kwadratowe” odwołuje się do procesu przesiewania (którego nie będziemy tu opisywać) używanego do znajdowania tych liczb  $x_j$ , które rozkładają się nad zbiorem  $\mathcal{B}$ .

Metoda wymaga pewnego kompromisu. Jeśli liczba  $B = |\mathcal{B}|$  jest duża, zwiększa się szansa na to, że liczba całkowita  $x_j$  rozkłada się nad zbiorem  $\mathcal{B}$ . Z drugiej strony, im większy jest ten zbiór, tym więcej potrzeba kongruencji, by uzyskać odpowiednie zależności. Optymalny rozmiar zbioru  $\mathcal{B}$  wyraża się w przybliżeniu liczbą

$$\sqrt{e^{\sqrt{\ln n \ln \ln n}}},$$

co daje oczekiwany czas działania algorytmu

$$O\left(e^{(1+o(1))\sqrt{\ln n \ln \ln n}}\right).$$

Sito ciała liczbowego jest algorymem nowszym, pochodzi bowiem z późnych lat osiemdziesiątych ubiegłego wieku. Rozkład na czynniki pierwsze liczby  $n$  uzyskuje się tu także przez konstrukcję kongruencji  $x^2 \equiv y^2 \pmod{n}$ , lecz obliczenia odbywają się w pierścieniu liczb całkowitych pewnego liczbowego ciała algebraicznego.

#### 4.8.3. Algorytmy faktoryzacji w praktyce

Asymptotyczne czasy działania sita kwadratowego, algorytmu krzywych eliptycznych oraz sita ciała liczbowego przedstawiają się następująco:

sito kwadratowe	$O\left(e^{(1+o(1))\sqrt{\ln n \ln \ln n}}\right)$
krzywe eliptyczne	$O\left(e^{(1+o(1))\sqrt{2 \ln p \ln \ln p}}\right)$
sito ciała liczbowego	$O\left(e^{(1,92+o(1))(\ln n)^{1/3}(\ln \ln n)^{2/3}}\right)$

Symbol  $o(1)$  oznacza tu funkcję zmiennej  $n$ , która dąży do 0, gdy  $n \rightarrow \infty$ , natomiast  $p$  jest najmniejszym czynnikiem pierwszym liczby  $n$ .

W najgorszym przypadku  $p \approx \sqrt{n}$ , a więc można przyjąć, że asymptotyczne czasy działania algorytmów sita kwadratowego i krzywych eliptycznych są równe, jednak w takich sytuacjach sito kwadratowe daje lepsze rezultaty. Z kolei metoda krzywych eliptycznych jest bardziej przydatna, gdy czynniki pierwsze liczby  $n$  różnią się znacząco wielkością. W 1988 roku Brent rozłożył tą metodą ogromną liczbę Fermata  $2^{2^{11}} - 1$ .

Do faktoryzacji modułów RSA (gdzie  $n = pq$ , a  $p$  i  $q$  są liczbami pierwszymi podobnej wielkości) najlepszym w tej chwili algorytmem jest sito kwadratowe. Jego rozwój znaczony jest kilkoma kamieniami milowymi. W 1983 roku Davis, Holdridge i Simmons rozłożyli na czynniki za pomocą sita kwadratowego liczbę 69-cyfrową, będącą (złożonym) czynnikiem liczby  $2^{251} - 1$ . W latach osiemdziesiątych następował dalszy postęp i w 1989 roku za pomocą tej metody Lenstra i Manasse rozkładali liczby zawierające do 106 cyfr, rozdzielając obliczenia między setki odległych od siebie stacji roboczych (ten tryb pracy nazwali „faktoryzacją przez pocztę elektroniczną”).

W czasach nieco bliższych, bo w kwietniu 1994 roku, Atkins, Graff, Lenstra i Leyland rozłożyli na czynniki metodą sita kwadratowego 129-cyfrową liczbę znaną jako RSA-129. (Liczby RSA-100, RSA-110, ..., RSA-500 znajdują się na liście modułów RSA opublikowanej w Internecie jako „wyzwanie” dla algorytmów faktoryzacji. Liczba RSA- $d$  jest  $d$ -cyfrowym iloczynem dwóch liczb pierwszych zbliżonej wielkości). Rozkład liczby RSA-129 wymagał 5000 mips-lat<sup>†</sup>, rozdzielonych między ok. 600 badaczy z całego świata.

Najnowszym z trzech omawianych algorytmów jest sito ciała liczbowego. Wydaje się on mieć wielki potencjał, jeśli sądzić po tym, że asymptotyczny czas jego działania jest krótszy zarówno od czasu działania sita kwadratowego, jak i metody krzywych eliptycznych. Znajduje się on wciąż na etapie opracowywania, jednak przypuszcza się, że może się on okazać szybszy dla liczb mających więcej niż 125–130 cyfr. W 1990 roku Lenstra, Lenstra, Manasse i Pollard użyli metody sita ciała liczbowego do rozłożenia liczby  $2^{2^9} - 1$  na trzy czynniki pierwsze o 7, 49 i 99 cyfrach.

## 4.9. Uwagi i bibliografia

Pomysł kryptografii z kluczem publicznym pochodzi od Diffiego i Hellmana (1976 r.). Choć najczęściej cytuję się [DH76A], nieco wcześniej ukazała się praca konferencyjna [DH76]. System kryptograficzny RSA zawdzięczamy Rivestowi, Shamirovi i Adlemanowi [RSA78]. System kryptograficzny Rabina został opisany przez Rabina w [RA79]; podobny system z jednoznacznym deszyfrowa-

<sup>†</sup>1 mips-rok to praca wykonana w ciągu roku przez komputer pracujący z szybkością 1 000 000 instrukcji na sekundę (*million-instructions-per-second*) (przyp. tłum.).

niem i udowodnionym bezpieczeństwem opracował Williams [Wi80]. Polecamy ogólny artykuł przeglądowy Diffiego [Di92], poświęcony kryptografi z kluczem publicznym.

Test Solovaya–Strassena został opisany po raz pierwszy w [SS77], test Millera–Rabina przedstawiono w [Mi76] i [Ra80]. Przy omawianiu prawdopodobieństwa błędu opieraliśmy się na obserwacjach Brassarda i Bratleya [BB88A, §8.6] (patrz także [BBCGP88]). Najlepsze obecnie oszacowania prawdopodobieństwa błędu dla algorytmu Millera–Rabina można znaleźć w [DLP93].

Materiał zawarty w podrozdziale 4.6 jest oparty na pracy Salomaa [Sa90, s.143–154]. Dowód faktoryzacji  $n$  przy danym wykładniku deszyfrowania znajduje się w [DE84]; wyniki o częściowej informacji, jakiej dostarcza RSA, pochodzą z [GMT82].

Jak wspomnieliśmy wcześniej, istnieje wiele źródeł wiedzy o algorytmach faktoryzacji. Pomerance [Po90] dokonał dobrego przeglądu problemu rozkładu na czynniki, natomiast udany artykuł Lenstry i Lenstry [LL90] zawiera ogólne omówienie algorytmów liczbowych. Książka Bressouda [Br89] stanowi elementarny podręcznik na temat faktoryzacji i sprawdzania pierwszości liczb. W kryptograficznych podręcznikach Koblitza [Ko95] i Kranakisa [Kr86] jest uwypuklona rola teorii liczb. Lenstra i Lenstra napisali monografię [LL93] poświęconą situa cia liczbowego.

Ćwiczenia 4.7–4.9 zawierają kilka przykładów niepowodzenia protokołu. Artykuł, w którym temat jest ujęty drobiazgowo, napisał Moore [Mo92].

## Ćwiczenia

4.1. Użyj rozszerzonego algorytmu Euklidesa do obliczenia następujących odwrotności:

- (a)  $17^{-1} \bmod 101$ ,
- (b)  $357^{-1} \bmod 1234$ ,
- (c)  $3125^{-1} \bmod 9987$ .

4.2. Rozwiąż następujący układ kongruencji:

$$\begin{aligned}x &\equiv 12 \pmod{25} \\x &\equiv 9 \pmod{26} \\x &\equiv 23 \pmod{27}.\end{aligned}$$

4.3. Rozwiąż następujący układ kongruencji:

$$\begin{aligned}13x &\equiv 4 \pmod{99} \\15x &\equiv 56 \pmod{101}.\end{aligned}$$

**WSKAZÓWKA** Użyj najpierw rozszerzonego algorytmu Euklidesa, a następnie zastosuj chińskie twierdzenie o resztach.

4.4. Zbadamy tu niektóre własności pierwiastków pierwotnych.

- Liczba 97 jest pierwsza. Wykaż, że  $x \neq 0$  jest pierwiastkiem pierwotnym modulo 97 wtedy i tylko wtedy, gdy  $x^{32} \not\equiv 1 \pmod{97}$  oraz  $x^{48} \not\equiv 1 \pmod{97}$ .
- Użyj tej metody do znalezienia najmniejszego pierwiastka pierwotnego modulo 97.
- Załóżmy, że liczba  $p$  jest pierwsza, a  $p - 1$  ma następujący rozkład na czynniki pierwsze:

$$p - 1 = \prod_{i=1}^n p_i^{e_i},$$

gdzie  $p_i$  są różnymi liczbami pierwszymi. Udowodnij, że  $x \neq 0$  jest pierwiastkiem pierwotnym modulo  $p$  wtedy i tylko wtedy, gdy  $x^{(p-1)/p_i} \not\equiv 1 \pmod{p}$  dla  $1 \leq i \leq n$ .

4.5. Niech  $n = pq$ , gdzie  $p$  i  $q$  są różnymi liczbami pierwszymi, oraz  $ab \equiv 1 \pmod{(p-1)(q-1)}$ . Określmy dla RSA operacje szyfrowania i deszyfrowania:  $e_K(x) = x^b \pmod{n}$ ,  $d_K(y) = y^a \pmod{n}$ . Wykazaliśmy, że  $d(e(x)) = x$ , gdy  $x \in \mathbb{Z}_n^*$ . Udowodnij tę własność dla dowolnego  $x \in \mathbb{Z}_n$ .

**WSKAZÓWKA** Z chińskiego twierdzenia o resztach wynika, że  $x_1 \equiv x_2 \pmod{pq}$  wtedy i tylko wtedy, gdy  $x_1 \equiv x_2 \pmod{p}$  i  $x_1 \equiv x_2 \pmod{q}$ .

4.6. W tablicach 4.1 i 4.2 są przedstawione dwie próbki tekstu zaszyfrowanego w RSA. Zadanie polega na ich odczytaniu. Publicznymi parametrami systemu są następujące wielkości:  $n = 18923$  i  $b = 1261$  (dla tablicy 1) oraz  $n = 31313$  i  $b = 4913$  (dla tablicy 2). Można je rozwiązać w sposób następujący. Po pierwsze, rozłóż na czynniki liczbę  $n$  (co nie sprawi trudności, jako że liczba  $n$  jest mała), a następnie, znając  $\phi(n)$ , oblicz wykładnik  $a$ . Potem odczytaj tekst zaszyfrowany. Do potęgowania modulo  $n$  użyj algorytmu podnoszenia do kwadratu i wymnażania.

Aby przetłumaczyć tekst jawny na język angielski, musisz wiedzieć, jak znaki alfabetu są „kodowane” w postaci elementów  $\mathbb{Z}_n$ . Każdy element  $\mathbb{Z}_n$  reprezentuje trzy znaki alfabetu, tak jak w następujących przykładach:

$$DOG \rightarrow 3 \cdot 26^2 + 14 \cdot 26 + 6 = 2398$$

$$CAT \rightarrow 2 \cdot 26^2 + 0 \cdot 26 + 19 = 1371$$

$$ZZZ \rightarrow 25 \cdot 26^2 + 25 \cdot 26 + 25 = 17575.$$

W ostatnim kroku Twojego programu będziesz musiał odwrócić tę procedurę.

Pierwszy tekst jawny pochodzi z „The Diary of Samuel Marchbanks” Robertsona Daviesa, 1947, drugi zaś z „Lake Wobegon Days” Garrisona Keillora, 1985.

4.7. To ćwiczenie ma ilustrować to, co określa się jako *niepowodzenie protokołu*. Stanoi ono przykład na to, że przeciwnik może odczytać tekst zaszyfrowany bez uprzedniego ustalenia klucza, jeśli system nie jest używany z należytą ostrożnością. (Ponieważ przeciwnik nie ustala klucza, trudno nazwać ten proces kryptanalizą). Wynika stąd morał, że nie wystarczy stosować „bezpieczny” system kryptograficzny, by zapewnić „bezpieczną” łączność.

TABLICA 4.1. Tekst zaszyfrowany za pomocą RSA

12423	11524	7243	7459	14303	6127	10964	16399
9792	13629	14407	18817	18830	13556	3159	16647
5300	13951	81	8986	8007	13167	10022	17213
2264	961	17459	4101	2999	14569	17183	15827
12693	9553	18194	3830	2664	13998	12501	18873
12161	13071	16900	7233	8270	17086	9792	14266
13236	5300	13951	8850	12129	6091	18110	3332
15061	12347	7817	7946	11675	13924	13892	18031
2620	6276	8500	201	8850	11178	16477	10161
3533	13842	7537	12259	18110	44	2364	15570
3460	9886	8687	4481	11231	7547	11383	17910
12867	13203	5102	4742	5053	15407	2976	9330
12192	56	2471	15334	841	13995	17592	13297
2430	9741	11675	424	6686	738	13874	8168
7913	6246	14301	1144	9056	15967	7328	13203
796	195	9872	16979	15404	14130	9105	2001
9792	14251	1498	11296	1105	4502	16979	1105
56	4118	11302	5988	3363	15827	6928	4191
4277	10617	874	13211	11821	3090	18110	44
2364	15570	3460	9886	9988	3798	1158	9872
16979	15404	6127	9872	3652	14838	7437	2540
1367	2512	14407	5053	1521	297	10935	17137
2186	9433	13293	7555	13618	13000	6490	5310
18676	4782	11374	446	4165	11634	3846	14611
2364	6789	11634	4493	4063	4576	17955	7965
11748	14616	11453	17666	925	56	4118	18031
9522	14838	7437	3880	11476	8305	5102	2999
18628	14326	9175	9061	650	18110	8720	15404
2951	722	15334	841	15610	2443	11056	2186

Przypuśćmy, że Bolek ma system kryptograficzny RSA z dużym modelem  $n$ , którego faktoryzacji nie można przeprowadzić w rozsądny czasie. Założymy ponadto, że Alicja, wysyłając do Bolka komunikat, reprezentuje każdy znak alfabetu za pomocą liczby całkowitej od 0 do 25 (czyli  $A \leftrightarrow 0, B \leftrightarrow 1$  itd.), po czym każdą resztę modulo 26 szyfruje jako odrębny znak tekstu jawnego.

- (a) Opisz, w jaki sposób Oskar może łatwo odczytać tak zaszyfrowany komunikat.
  - (b) Zilustruj ten rodzaj ataku, deszyfrując następujący tekst zaszyfrowany (za pomocą systemu RSA z parametrami  $n = 18721$  i  $b = 25$ ) bez rozkładania modułu na czynniki:
- 365, 0, 4845, 14930, 2608, 2608, 0.
- 4.8. To ćwiczenie ilustruje inny (opisany przez Simmonsa) związek z RSA przykład niepowodzenia protokołu, nazywany niepowodzeniem *wspólnego modułu*. Założymy, że Bolek dysponuje systemem RSA z modelem  $n$  i wykładownikiem deszyfrowania  $b_1$ , natomiast Cezary dla tego samego systemu używa modułu  $n$  (tego samego co Bolek) i wykładownika deszyfrowania  $b_2$ . Przyjmijmy ponadto, że

TABLICA 4.2. Tekst zaszyfrowany za pomocą RSA

6340	8309	14010	8936	27358	25023	16481	25809
23614	7135	24996	30590	27570	26486	30388	9395
27584	14999	4517	12146	29421	26439	1606	17881
25774	7647	23901	7372	25774	18436	12056	13547
7908	8635	2149	1908	22076	7372	8686	1304
4082	11803	5314	107	7359	22470	7372	22827
15698	30317	4685	14696	30388	8671	29956	15705
1417	26905	25809	28347	26277	7897	20240	21519
12437	1108	27106	18743	24144	10685	25234	30155
23005	8267	9917	7994	9694	2149	10042	27705
15930	29748	8635	23645	11738	24591	20240	27212
27486	9741	2149	29329	2149	5501	14015	30155
18154	22319	27705	20321	23254	13624	3249	5443
2149	16975	16087	14600	27705	19386	7325	26277
19554	23614	7553	4734	8091	23973	14015	107
3183	17347	25234	4595	21498	6360	19837	8463
6000	31280	29413	2066	369	23204	8425	7792
25973	4477	30989					

NWD( $b_1, b_2$ ) = 1. Rozpatrzmy teraz sytuację, która powstanie, gdy Alicja zechce przekazać szyfrem ten sam tekst jawnego do Bolka i Cezarego. Wówczas oblicza  $y_1 = x^{b_1} \bmod n$  oraz  $y_2 = x^{b_2} \bmod n$ , po czym wysyła  $y_1$  do Bolka oraz  $y_2$  do Cezarego. Oskar przejmuje oba teksty  $y_1$  i  $y_2$  i wykonuje obliczenia pokazane na rysunku 4.16.

- (a) Wykaż, że wartość  $x_1$  obliczona w kroku 3 algorytmu z rysunku 4.16 jest w istocie tekstem jawnym Alicji, czyli  $x$ . Tak więc Oskar może odczytać komunikat Alicji, nawet jeśli system kryptograficzny jest „bezpieczny”.
- (b) Zilustruj atak, obliczając tą metodą  $x$ , gdy  $n = 18721$ ,  $b_1 = 43$ ,  $b_2 = 7717$ ,  $y_1 = 12677$  oraz  $y_2 = 14702$ .

Dane wejściowe: $n, b_1, b_2, y_1, y_2$
(1) oblicz $c_1 = b_1^{-1} \bmod b_2$
(2) oblicz $c_2 = (c_1 b_1 - 1)/b_2$
(3) oblicz $x_1 = y_1^{c_1} (y_2^{c_2})^{-1} \bmod n$

RYSUNEK 4.16. Niepowodzenie protokołu RSA (niepowodzenie wspólnego modułu)

- 4.9. Oto jeszcze jedno niepowodzenie protokołu związanego z RSA. Założymy, że mamy trzech użytkowników w sieci (powiedzmy Bolek, Bartek i Bernard). Wszyscy mają publiczny wykładnik deszyfrowania  $b = 3$ . Niech ich modułami będą liczby  $n_1, n_2, n_3$ . Przypuśćmy teraz, że Alicja szyfruje ten sam tekst jawnego  $x$  dla Bolka, Bartka i Bernarda, czyli oblicza  $y_i = x^3 \bmod n_i$ ,  $1 \leq i \leq 3$ . Opisz, w jaki sposób, mając dane teksty  $y_1, y_2, y_3$ , Oskar może obliczyć  $x$  bez rozkładania na czynniki jakiegokolwiek modułu.

- 4.10. Mówimy, że tekst jawny jest *stały*, gdy  $e_K(x) = x$ . Wykaż, że dla systemu kryptograficznego RSA liczba stałych tekstów jawnych  $x \in \mathbb{Z}_n^*$  jest równa  $\text{NWD}(b-1, p-1) \cdot \text{NWD}(b-1, q-1)$ .

**WSKAZÓWKA** Rozważ układ dwóch kongruencji:  $e_K(x) \equiv x \pmod{p}$ ,  $e_K(x) \equiv x \pmod{q}$ .

- 4.11. Niech **A** będzie algorytmem deterministycznym przyjmującym jako dane wejściowe moduł RSA  $n$ , wykładownik szyfrowania  $b$  oraz tekst zaszyfrowany  $y$ . W wyniku działania **A** albo podaje wynik deszyfrowania tekstu  $y$ , albo nie udziela żadnej odpowiedzi. Przyjmując, że istnieje  $\epsilon(n-1)$  tekstów zaszyfrowanych, które **A** potrafi odczytać, opisz jak użyć **A** jako wyróczni w algorytmie deszyfrowania typu Las Vegas z prawdopodobieństwem sukcesu  $\epsilon$ .

**WSKAZÓWKA** Wykorzystaj zgodność operacji szyfrowania z mnożeniem:  $e_K(x_1)e_K(x_2) = e_K(x_1x_2)$ , gdzie wszystkie działania arytmetyczne są wykonywane modulo  $n$ .

- 4.12. Napisz program do obliczania symboli Jacobiego, który używałby czterech właściwości opisanych w podrozdziale 4.5. Program nie powinien przeprowadzać żadnej faktoryzacji prócz dzielenia potęg liczby 2. Przetestuj swój program na następujących symbolach Jacobiego:

$$\left(\frac{610}{987}\right), \left(\frac{20964}{1987}\right), \left(\frac{1234567}{11111111}\right).$$

- 4.13. Dla  $n = 837, 851$  i  $1189$  określ liczbę podstaw  $b$ , takich że  $n$  jest liczbą pseudopierwszą Eulera przy podstawie  $b$ .
- 4.14. W tym ćwiczeniu udowodnimy, że prawdopodobieństwo błędu w teście pierwszości Solovaya-Strassena nie przekracza  $1/2$ . Niech  $\mathbb{Z}_n^*$  oznacza grupę jedności modulo  $n$ . Niech

$$G(n) = \left\{ a : a \in \mathbb{Z}_n^*, \left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \pmod{n} \right\}.$$

- (a) Wykaż, że  $G(n)$  jest podgrupą grupy  $\mathbb{Z}_n^*$ . Stąd, na mocy twierdzenia Lagrange'a, jeśli  $G(n) \neq \mathbb{Z}_n^*$ , to

$$|G(n)| \leq \frac{|\mathbb{Z}_n^*|}{2} \leq \frac{n-1}{2}.$$

- (b) Założmy, że  $n = p^k q$ , gdzie  $p$  i  $q$  są liczbami nieparzystymi,  $p$  jest pierwsza,  $k \geq 2$  oraz  $\text{NWD}(p, q) = 1$ . Niech  $a = 1 + p^{k-1}q$ . Udowodnij, że

$$\left(\frac{a}{n}\right) \not\equiv a^{(n-1)/2} \pmod{n}.$$

**WSKAZÓWKA** Użyj wzoru dwumianowego do obliczenia  $a^{(n-1)/2}$ .

- (c) Niech  $n = p_1 \dots p_s$ , gdzie  $p_i$  są różnymi nieparzystymi liczbami pierwszymi. Niech  $a \equiv u \pmod{p_1}$  oraz  $a \equiv 1 \pmod{p_2 p_3 \dots p_n}$ , gdzie  $u$  jest nierówną kwadratową modulo  $p_1$  (zauważ, że takie  $a$  istnieje na mocy chińskiego twierdzenia o resztach). Wykaż, że

$$\left(\frac{a}{n}\right) \equiv -1 \pmod{n},$$

lecz

$$a^{(n-1)/2} \equiv 1 \pmod{p_2 p_3 \dots p_s},$$

zatem

$$a^{(n-1)/2} \not\equiv -1 \pmod{n}.$$

- (d) Wykaż, że jeśli  $n$  jest liczbą nieparzystą i złożoną, to  $|G(n)| \leq (n-1)/2$ .
- (e) Podsumuj dotychczasowe wyniki: udowodnij, że prawdopodobieństwo błędu w teście pierwszości Solovaya–Strassena nie przekracza  $1/2$ .
- 4.15. Założymy, że mamy algorytm typu Las Vegas z prawdopodobieństwem niepowodzenia równym  $\epsilon$ .
- (a) Udowodnij, że prawdopodobieństwo uzyskania pierwszego sukcesu w  $n$ -tej próbie jest równe  $p_n = \epsilon^{n-1}(1-\epsilon)$ .
- (a) Średnia (oczekiwana) liczba prób potrzebna do osiągnięcia sukcesu jest równa
- $$\sum_{n=1}^{\infty} (n \cdot p_n).$$
- Wykaż, że ta średnia jest równa  $1/(1-\epsilon)$ .
- (b) Niech  $\delta$  będzie dodatnią liczbą rzeczywistą mniejszą od 1. Udowodnij, że liczba iteracji potrzebnych do ograniczenia prawdopodobieństwa niepowodzenia do co najwyżej  $\delta$  wynosi
- $$\left\lceil \frac{\log_2 \delta}{\log_2 \epsilon} \right\rceil.$$
- 4.16. Przypuśćmy, że Bolek nieopatrznie ujawnił swój wykładnik deszyfrowania  $a = 14039$  w systemie kryptograficznym RSA z kluczem publicznym  $n = 36581$  i  $b = 4679$ . Zastosuj algorytm probabilistyczny do faktoryzacji  $n$  na podstawie tej informacji. Sprawdź swój algorytm na „losowo” wybranych wartościach  $w = 9983$  i  $w = 13461$ . Przedstaw wszystkie obliczenia.
- 4.17. Udowodnij równości (4.1) i (4.2), które wiążą funkcje *pol* i *par*.
- 4.18. Założymy, że w systemie Rabina  $p = 199$ ,  $q = 211$  oraz  $B = 1357$ . Wykonaj następujące obliczenia:
- (a) Wyznacz cztery pierwiastki kwadratowe z 1 modulo  $n$ , jeśli  $n = pq$ .
- (b) Oblicz tekst zaszyfrowany  $y = e_K(32767)$ .
- (c) Wyznacz cztery możliwe wyniki deszyfrowania tekstu zaszyfrowanego  $y$ .
- 4.19. Rozłóż na czynniki pierwsze liczby 262063 i 9420457 za pomocą metody  $p-1$ . Jak duże musi być  $B$  w każdym z tych przypadków, by zapewnić sukces?

# 5

---

## Inne systemy kryptograficzne z kluczem publicznym

---

W tym rozdziale omówimy kilka innych systemów kryptograficznych z kluczem publicznym. System ElGamala opiera się na problemie logarytmu dyskretnego, którego będziemy mieli okazję używać wielokrotnie przy protokołach kryptograficznych w pozostałej części tekstu i dlatego poświęcimy mu tu wiele miejsca. Następnie, przedstawimy stosunkowo krótkie omówienia niektórych innych dobrze znanych systemów z kluczem publicznym. Są wśród nich systemy typu ElGamala, oparte na teorii ciał skończonych i krzywych eliptycznych: (złamany) plecakowy system kryptograficzny Merkle'a–Hellmana oraz system kryptograficzny McEliece'a.

---

### 5.1. System kryptograficzny ElGamala i logarytmy dyskretne

Podstawą systemu **ElGamala** jest problem **logarytmu dyskretnego**. Zaczniemy więc od przedstawienia tego problemu na rysunku 5.1 w kontekście ciała skońzonego  $\mathbb{Z}_p$ , gdzie  $p$  jest liczbą pierwszą. (Przypomnijmy, że grupa mnożenia  $\mathbb{Z}_p^*$  jest cykliczna, a generator tej grupy nazywamy elementem pierwotnym).

**Dane**  $I = (p, \alpha, \beta)$ , gdzie  $p$  jest liczbą pierwszą,  $\alpha \in \mathbb{Z}_p$  jest elementem pierwotnym oraz  $\beta \in \mathbb{Z}_p^*$ .

**Cel** Znaleźć jedyną liczbę całkowitą  $a$ ,  $0 \leq a \leq p - 2$ , taką że

$$\alpha^a \equiv \beta \pmod{p}.$$

Taką liczbę  $a$  oznaczymy symbolem  $\log_{\alpha} \beta$ .

RYSUNEK 5.1. Problem logarytmu dyskretnego w  $\mathbb{Z}_p$

Problem logarytmu dyskretnego w  $\mathbb{Z}_p$  był przedmiotem wielu badań. Uważa się powszechnie, iż przy starannym doborze liczby  $p$  jest to problem trudny. W szczególności nie jest znany żaden algorytm rozwiązuający problem logarytmu dyskretnego w czasie wielomianowym. Aby zniweczyć wszelkie znane sposoby ataku, liczba  $p$  powinna mieć co najmniej 150 cyfr, a co najmniej jednym z czynników liczby  $p - 1$  powinna być „duża” liczba pierwsza. Użyteczność problemu logarytmu dyskretnego w kryptografii bierze się z tego, że znajdowanie dyskretnych logarytmów jest (prawdopodobnie) trudne, natomiast odwrotna operacja potęgowania jest obliczeniowo łatwa, gdy zastosuje się opisaną wcześniej metodę podnoszenia do kwadratu i wymnażania. Inaczej mówiąc, funkcja potęgowania modulo  $p$  jest dla odpowiednich liczb pierwszych funkcją jednokierunkową.

ElGamal opracował system kryptograficzny z kluczem publicznym oparty na problemie logarytmu dyskretnego. Przedstawiamy go na rysunku 5.2.

Niech  $p$  będzie liczbą pierwszą, taka że problem logarytmu dyskretnego jest obliczeniowo nieroziągalny w  $\mathbb{Z}_p$ , i niech  $\alpha \in \mathbb{Z}_a[p]$  będzie elementem pierwotnym. Przyjmijmy  $\mathcal{P} = \mathbb{Z}_p^*$ ,  $\mathcal{C} = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ . Określmy

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}.$$

Wartości  $p$ ,  $\alpha$  oraz  $\beta$  są znane publicznie, wartość  $a$  jest tajna.

Dla  $\mathcal{K} = (p, \alpha, a, \beta)$  i (tajnej) losowej liczby  $k \in \mathbb{Z}_{p-1}$  definiujemy

$$e_K(x, k) = (y_1, y_2),$$

gdzie

$$y_1 = \alpha^k \pmod{p}$$

oraz

$$y_2 = x\beta^k \pmod{p}.$$

Dla  $y_1, y_2 \in \mathbb{Z}_p^*$  definiujemy

$$d_K(y_1, y_2) = y_2(y_1^a)^{-1} \pmod{p}.$$

**RYSUNEK 5.2.** System kryptograficzny ElGamala z kluczem publicznym w  $\mathbb{Z}_p^*$

System ElGamala jest systemem niedeterministycznym, ponieważ tekst zaszyfrowany zależy nie tylko od tekstu jawnego, ale także od losowej wartości  $k$  wybranej przez Alicję. Dlatego ten sam tekst jawnny może być zaszyfrowany na wiele różnych sposobów.

Spróbujmy opisać nieformalnie działanie systemu ElGamala. Tekst jawny  $x$  jest najpierw „maskowany” przez pomnożenie przez  $\beta^k$ , czego efektem jest tekst  $y_2$ . W ramach tekstu zaszyfrowanego przekazuje się także wartość  $\alpha^k$ . Bolek, który zna tajny wykładnik  $a$ , może na podstawie znajomości  $\alpha^k$  obliczyć  $\beta^k$ , a następnie usunąć „maskę”, dzieląc  $y_2$  przez  $\beta^k$ . W ten sposób otrzymuje tekst jawny  $x$ .

Popatrzmy na prosty przykład.

### Przykład 5.1

Niech  $p = 2579$ ,  $\alpha = 2$ ,  $a = 765$ , a zatem

$$\begin{aligned}\beta &= 2^{765} \pmod{2579} \\ &= 949.\end{aligned}$$

Założymy teraz, że Alicja chce przesłać Bolkowi komunikat  $x = 1299$ . Niech losowo wybraną przez nią liczbą całkowitą będzie  $k = 853$ . Oto jej obliczenia:

$$\begin{aligned}y_1 &= 2^{853} \pmod{2579} \\ &= 435\end{aligned}$$

oraz

$$\begin{aligned}y_2 &= 1299 \cdot 949^{853} \pmod{2579} \\ &= 2396.\end{aligned}$$

Gdy Bolek dostaje tekst zaszyfrowany  $y = (435, 2396)$ , oblicza  $x$ :

$$\begin{aligned}x &= 2396 \cdot (435^{765})^{-1} \pmod{2579} \\ &= 1299,\end{aligned}$$

a więc rzeczywiście otrzymuje tekst zaszyfrowany przez Alicję. □

#### 5.1.1. Algorytmy dla problemu logarytmu dyskretnego

Zakładamy w tym podrozdziale, że  $p$  jest ustaloną liczbą pierwszą, a  $\alpha$  ustalonym elementem pierwotnym modulo  $p$ . Problem **logarytmu dyskretnego** możemy teraz sformułować w sposób następujący: dla danego  $\beta \in \mathbb{Z}_p^*$  należy znaleźć jedyny wykładnik  $a$ , taki że  $0 \leq a \leq p - 2$  oraz  $\alpha^a \equiv \beta \pmod{p}$ .

Oczywiście, problem logarytmu dyskretnego można rozwiązać metodą wyyczerpującego przeszukiwania w czasie  $O(p)$  i z pamięcią  $O(1)$  (z pominięciem czynników logarytmicznych). Jeśli w obliczeniach wstępnych znajdziemy wszystkie możliwe wartości  $\alpha^a$  i posortujemy wszystkie pary uporządkowane  $(a, \alpha^a \pmod{p})$  ze względu na drugie współrzędne, to problem logarytmu dyskretnego możemy rozwiązać w czasie  $O(1)$  z czasem obliczeń wstępnych  $O(p)$  oraz

pamięcią  $O(p)$  (znów z pominięciem czynników logarytmicznych). Pierwszym nietrywialnym algorytmem, który tu opiszemy, jest algorytm Shanksa, oferujący kompromis między czasem i pamięcią.

### Algorytm Shanksa

Niech  $m = \lceil \sqrt{p-1} \rceil$ . Algorytm Shanksa przedstawiamy na rysunku 5.3, tu warto dodać kilka uwag. Po pierwsze, kroki 1 i 2 można, jeśli się chce, wykonać w ramach obliczeń wstępnych (choć nie wpłynie to na asymptotyczny czas działania algorytmu). Dalej, zauważmy, że jeśli  $(j, y) \in L_1$  oraz  $(i, y) \in L_2$ , to

$$\begin{aligned}\alpha^{mj} &= y \\ &= \beta\alpha^{-i},\end{aligned}$$

zatem

$$\alpha^{mj+i} = \beta,$$

zgodnie z oczekiwaniami. Na odwrót, dla dowolnego  $\beta$  możemy napisać

$$\log_a \beta = mj + i,$$

gdzie  $0 \leq j, i \leq m-1$ . Tak więc wyszukiwanie w kroku 5 zakończy się sukcesem.

- (1) oblicz  $\alpha^{mj} \pmod p$ ,  $0 \leq j \leq m-1$
- (2) posortuj  $m$  par uporządkowanych  $(j, \alpha^{mj} \pmod p)$  ze względu na drugą współrzędną; niech  $L_1$  będzie otrzymaną listą
- (3) oblicz  $\beta\alpha^{-1} \pmod p$ ,  $0 \leq i \leq m-1$
- (4) posortuj  $m$  par uporządkowanych  $(i, \beta\alpha^{-1} \pmod p)$  ze względu na drugą współrzędną; niech  $L_2$  będzie otrzymaną listą
- (5) znajdź parę  $(j, y) \in L_1$  oraz parę  $(i, y) \in L_2$  (czyli pary o tej samej drugiej współrzędnej)
- (6) określ  $\log_a \beta = mj + i \pmod{(p-1)}$

**RYSUNEK 5.3.** Algorytm Shanksa dla problemu logarytmu dyskretnego

Nietrudno tak zrealizować algorytm, by działał w czasie  $O(m)$  i z pamięcią  $O(m)$  (z pominięciem czynników logarytmicznych). Zauważmy, że krok 5 można wykonać jednym (jednoczesnym) przejściem każdej z list  $L_1$  i  $L_2$ . Oto przykład dla ilustracji.

#### Przykład 5.2

Niech  $p = 809$ . Założymy, że chcemy znaleźć  $\log_3 525$ . Mamy zatem  $\alpha = 3$ ,  $\beta = 525$  oraz  $m = \lceil \sqrt{808} \rceil = 29$ . Wówczas

$$\alpha^{29} \pmod{809} = 99.$$

Znajdziemy najpierw pary uporządkowane  $(j, 99^j \bmod 809)$  dla  $0 \leq j \leq 28$ . Otrzymujemy następującą listę:

(0, 1)	(1, 99)	(2, 93)	(3, 308)	(4, 559)
(5, 329)	(6, 211)	(7, 664)	(8, 207)	(9, 268)
(10, 644)	(11, 654)	(12, 26)	(13, 147)	(14, 800)
(15, 727)	(16, 781)	(17, 464)	(18, 632)	(19, 275)
(20, 528)	(21, 496)	(22, 564)	(23, 15)	(24, 676)
(25, 586)	(26, 575)	(27, 295)	(28, 81)	

Po posortowaniu listy powstaje  $L_1$ .

Druga lista zawiera pary uporządkowane  $(i, 525 \cdot (3^i)^{-1} \bmod 809)$ ,  $0 \leq i \leq 28$ :

(0, 525)	(1, 175)	(2, 328)	(3, 379)	(4, 396)
(5, 132)	(6, 44)	(7, 554)	(8, 724)	(9, 511)
(10, 440)	(11, 686)	(12, 768)	(13, 256)	(14, 355)
(15, 388)	(16, 399)	(17, 133)	(18, 314)	(19, 644)
(20, 754)	(21, 521)	(22, 713)	(23, 777)	(24, 259)
(25, 356)	(26, 658)	(27, 489)	(28, 163)	

Po posortowaniu otrzymujemy listę  $L_2$ .

Przechodząc teraz równocześnie obie posortowane listy, stwierdzamy, że (10, 644) znajduje się na liście  $L_1$ , a para (19, 644) na liście  $L_2$ . Obliczamy zatem:

$$\begin{aligned}\log_3 525 &= 29 \cdot 10 + 19 \\ &= 309.\end{aligned}$$

Teraz możemy sprawdzić, że istotnie,  $3^{309} \equiv 525 \pmod{809}$ . □

### Algorytm Pohliga–Hellmana

Przyjrzymy się teraz algorytmowi Pohliga–Hellmana. Niech

$$p - 1 = \prod_{i=1}^k p_i^{c_i},$$

gdzie  $p_i$  są różnymi liczbami pierwszymi. Wartość  $a = \log_\alpha \beta$  jest określona (jednoznacznie) modulo  $p - 1$ . Zauważmy od razu, że jeśli możemy obliczyć  $a \pmod{p_i^{c_i}}$  dla każdego  $i$ , takiego że  $1 \leq i \leq k$ , to dzięki chińskiemu twierdzeniu o resztach możemy także obliczyć  $a \pmod{(p - 1)}$ . Założymy zatem, że  $q$  jest liczbą pierwszą,

$$p - 1 \equiv 0 \pmod{q^c}$$

oraz

$$p - 1 \not\equiv 0 \pmod{q^{c+1}}.$$

Pokażemy, jak obliczyć wartość

$$x = a \bmod q^c,$$

gdzie  $0 \leq x \leq q^c - 1$ . Możemy zapisać  $x$  w układzie liczbowym o podstawie  $q$ :

$$x = \sum_{i=0}^{c-1} a_i q^i,$$

gdzie  $0 \leq a_i \leq q - 1$  dla  $0 \leq i \leq c - 1$ . Zauważmy także, że liczbę  $a$  możemy przedstawić w postaci

$$a = x + q^c s$$

dla pewnej liczby całkowitej  $s$ .

Pierwszy krok algorytmu polega na obliczeniu  $a_0$ . Kluczem do rozwiązania jest stwierdzenie następującej własności:

$$\beta^{(p-1)/q} \equiv \alpha^{(p-1)a_0/q} \pmod{p}.$$

Aby się przekonać o jej prawdziwości, zauważmy, że

$$\beta^{(p-1)/q} \equiv \alpha^{(p-1)(x+q^c s)/q} \pmod{p}.$$

Wystarczy więc wykazać, że

$$\alpha^{(p-1)(x+q^c s)/q} \equiv \alpha^{(p-1)a_0/q} \pmod{p}.$$

Ten ostatni warunek jest równoważny następującemu:

$$\frac{(p-1)(x+q^c s)}{q} \equiv \frac{(p-1)a_0}{q} \pmod{p-1}.$$

Jednakże mamy

$$\begin{aligned} \frac{(p-1)(x+q^c s)}{q} - \frac{(p-1)a_0}{q} &= \frac{(p-1)}{q}(x+q^c s - a_0) \\ &= \frac{(p-1)}{q} \left( \sum_{i=0}^{c-1} a_i q^i + q^c s - a_0 \right) \\ &= \frac{(p-1)}{q} \left( \sum_{i=1}^{c-1} a_i q^i + q^c s \right) \\ &= (p-1) \left( \sum_{i=1}^{c-1} a_i q^{i-1} + q^{c-1} s \right) \\ &\equiv 0 \pmod{p-1}, \end{aligned}$$

co było do udowodnienia.

Możemy zatem zacząć od obliczenia  $\beta^{(p-1)/q} \pmod{p}$ . Jeśli  $\beta^{(p-1)/q} \equiv 1 \pmod{p}$ ,

to  $a_0 = 0$ . W przeciwnym razie obliczamy kolejno

$$\gamma = \alpha^{(p-1)/q} \pmod{p}, \gamma^2 \pmod{p}, \dots,$$

aż do uzyskania

$$\gamma^i \equiv \beta^{(p-1)/q} \pmod{p}$$

dla pewnego  $i$ . Wtedy  $a_0 = i$ .

Jeśli  $c = 1$ , zadanie jest zakończone. W przeciwnym razie  $c > 1$  i przystępujemy do wyznaczenia  $a_1$ . W tym celu przyjmijmy

$$\beta_1 = \beta\alpha^{-a_0}$$

i oznaczmy

$$x_1 = \log_{\alpha} \beta_1 \pmod{q^c}.$$

Nietrudno zauważyć, że

$$x_1 = \sum_{i=1}^{c-1} a_i q^i.$$

Wynika stąd, że

$$\beta_1^{(p-1)/q^2} \equiv \alpha^{(p-1)a_1/q} \pmod{p}.$$

Obliczymy zatem  $\beta_1^{(p-1)/q^2} \pmod{p}$ , po czym znajdziemy  $i$ , takie że

$$\gamma^i \equiv \beta_1^{(p-1)/q^2} \pmod{p}.$$

Wówczas stwierdzimy, że  $a_1 = i$ .

Jeśli  $c = 2$ , proces jest zakończony. W przeciwnym razie powtarzamy go jeszcze  $c - 2$  razy, otrzymując kolejno  $a_2, \dots, a_{c-1}$ .

Opis algorytmu Pohliga–Hellmana za pomocą pseudokodu jest przedstawiony na rysunku 5.4. Zakładamy, że  $\alpha$  jest elementem pierwotnym modulo  $p$ ,  $q$  jest liczbą pierwszą,

$$p - 1 \equiv 0 \pmod{q^c}$$

oraz

$$p - 1 \not\equiv 0 \pmod{q^{c+1}}.$$

Algorytm oblicza  $a_0, \dots, a_{c-1}$ , gdzie

$$\log_{\alpha} \beta \pmod{q^c} = \sum_{i=0}^{c-1} a_i q^i.$$

Zobaczmy na prostym przykładzie, jak działa ten algorytm.

- (1) oblicz  $\gamma_i = \alpha^{(p-1)i/q} \pmod{p}$  dla  $0 \leq i \leq q-1$
- (2) przyjmij  $j = 0$  i  $\beta_j = \beta$
- (3) **while**  $j \leq c-1$  **do**
- (4)     oblicz  $\delta = \beta_j^{(p-1)/q^{j+1}} \pmod{p}$
- (5)     znajdź  $i$ , takie że  $\delta = \gamma_i$
- (6)      $a_j = i$
- (7)      $\beta_{j+1} = \beta_j \alpha^{-a_j q^j} \pmod{p}$
- (8)      $j = j + 1$

RYSUNEK 5.4. Algorytm Pohliga-Hellmana do obliczania  $\log_\alpha \beta \pmod{q^c}$ *Przykład 5.3*Niech  $p = 29$ . Wtedy

$$n = p - 1 = 28 = 2^2 7^1.$$

Załóżmy, że  $\alpha = 2$  i  $\beta = 18$ ; chcemy zatem wyznaczyć  $a = \log_2 18$ . Zaczniemy od obliczenia  $a \pmod{4}$ , po czym obliczymy  $a \pmod{7}$ .

Na początek ustalmy  $q = 2$  oraz  $c = 2$ . Po pierwsze,

$$\gamma_0 = 1$$

oraz

$$\begin{aligned}\gamma_1 &= \alpha^{28/2} \pmod{29} \\ &= 2^{14} \pmod{29} \\ &= 28.\end{aligned}$$

Następnie,

$$\begin{aligned}\delta &= \beta^{28/2} \pmod{29} \\ &= 18^{14} \pmod{29} \\ &= 28.\end{aligned}$$

Tak więc  $a_0 = 1$ . Obliczamy dalej:

$$\begin{aligned}\beta_1 &= \beta_0 \alpha^{-1} \pmod{29} \\ &= 9\end{aligned}$$

oraz

$$\begin{aligned}\beta_1^{28/4} \pmod{29} &= 9^7 \pmod{29} \\ &= 28.\end{aligned}$$

Z kongruencji

$$\gamma_1 \equiv 28 \pmod{29}$$

wnosimy, że  $a_1 = 1$ . W rezultacie mamy  $a \equiv 3 \pmod{4}$ .

Teraz przyjmujemy  $q = 7$  i  $c = 1$ . Mamy wtedy

$$\begin{aligned}\beta^{28/7} \bmod 29 &= 18^4 \bmod 29 \\ &= 25\end{aligned}$$

oraz

$$\begin{aligned}\gamma_1 &= \alpha^{28/7} \bmod 29 \\ &= 2^4 \bmod 29 \\ &= 16.\end{aligned}$$

W kolejnych obliczeniach uzyskujemy

$$\begin{aligned}\gamma_2 &= 24 \\ \gamma_3 &= 7 \\ \gamma_4 &= 25.\end{aligned}$$

Tak więc  $a_0 = 4$  i  $a \equiv 4 \pmod{7}$ .

Na koniec, za pomocą chińskiego twierdzenia o resztach, rozwiążujemy układ

$$\begin{aligned}a &\equiv 3 \pmod{4} \\ a &\equiv 4 \pmod{7},\end{aligned}$$

otrzymując  $a \equiv 11 \pmod{28}$ . Okazuje się więc, że  $\log_2 18$  jest w  $\mathbb{Z}_{29}$  równy 11. □

### Metoda obliczania indeksu

Metoda wyznaczania logarytmu dyskretnego poprzez obliczanie indeksu jest bardzo podobna do wielu spośród najlepszych algorytmów rozkładu na czynniki pierwsze. Przedstawimy tu jej bardzo zwięzły opis. Korzysta się w niej z *bazy rozkładu*, którą jest, jak poprzednio, zbiór  $\mathcal{B}$  „małych” liczb pierwszych. Niech  $\mathcal{B} = \{p_1, p_2, \dots, p_B\}$ . Pierwszy krok (krok wstępny) polega na wyznaczeniu logarytmów liczb pierwszych ze zbioru  $\mathcal{B}$ . W drugim kroku na podstawie tej wiedzy oblicza się logarytm dyskretny żądanego elementu  $\beta$ .

Na etapie wstępny tworzymy  $C = B + 10$  kongruencji modulo  $p$ :

$$\alpha^{x_j} \equiv p_1^{a_{1j}} p_2^{a_{2j}} \dots p_B^{a_{Bj}} \pmod{p}$$

dla  $1 \leq j \leq C$ . Zauważmy, że możemy te kongruencje zapisać w postaci równoważnej:

$$x_j \equiv a_{1j} \log_\alpha p_1 + \dots + a_{Bj} \log_\alpha p_B \pmod{p-1},$$

$i \leq j \leq C$ . Mając  $C$  kongruencji z  $B$  „niewiadomymi”  $\log_\alpha p_i$  ( $1 \leq i \leq B$ ), liczymy na uzyskanie jednoznacznego rozwiązania modulo  $p-1$ . Jeśli takie rozwiązanie znajdziemy, potrafimy obliczyć logarytmy liczb należących do bazy.

Jak zbudować kongruencje żądanej postaci? Jeden z elementarnych sposobów polega na wzięciu losowej wartości  $x$ , obliczeniu  $\alpha^x \bmod p$ , a następnie ustaleniu, czy wszystkie czynniki pierwsze liczby  $\alpha^x \bmod p$  należą do zbioru  $\mathcal{B}$  (np. wykonując dzielenie próbne).

Zakładając, że zakończyliśmy pomyślnie etap wstępny, przystępujemy dalej do obliczenia żądanego logarytmu  $\log_\alpha \beta$ . Stosujemy w tym celu algorytm probabilistyczny typu Las Vegas. Wybieramy losową liczbę całkowitą  $s$  (taką że  $1 \leq s \leq p - 2$ ) i obliczamy

$$\gamma = \beta \alpha^s \bmod p.$$

Teraz próbujemy rozłożyć liczbę  $\gamma$  na czynniki należące do  $\mathcal{B}$ . Gdy już to się uda, otrzymujemy kongruencję postaci

$$\beta \alpha^s \equiv p_1^{c_1} p_2^{c_2} \dots p_B^{c_B} \pmod{p},$$

która możemy zapisać równoważnie

$$\log_\alpha \beta + s \equiv c_1 \log_\alpha p_1 + \dots + c_B \log_\alpha p_B \pmod{p}.$$

Znamy wszystkie występujące tu liczby prócz  $\log_\alpha \beta$ , możemy zatem tę ostatnią wartość bez trudu obliczyć.

Oto prosty, bardzo sztuczny przykład, ilustrujący opisane wyżej dwa kroki algorytmu.

#### Przykład 5.4

Założymy, że  $p = 10007$ , a  $\alpha = 5$  jest elementem pierwotnym użytym jako podstawa logarytmów modulo  $p$ . Niech bazą rozkładu będzie zbiór  $\mathcal{B} = \{2, 3, 5, 7\}$ . Oczywiście  $\log_5 5 = 1$ , pozostałe zatem wyznaczyć pozostałe trzy logarytmy elementów z bazy.

Za przykład „szczęśliwie” wybranych wykładników niech posłużą liczby 4063, 5136 oraz 9865.

Dla  $x = 4063$  obliczamy

$$\begin{aligned} 5^{4063} \bmod 10007 &= 42 \\ &= 2 \cdot 3 \cdot 7. \end{aligned}$$

Otrzymujemy stąd kongruencję

$$\log_5 2 + \log_5 3 + \log_5 7 \equiv 4063 \pmod{10006}.$$

Podobnie z równości

$$\begin{aligned} 5^{5136} \bmod 10007 &= 54 \\ &= 2 \cdot 3^3 \end{aligned}$$

oraz

$$\begin{aligned} 5^{9865} \bmod 10007 &= 189 \\ &= 3^3 \cdot 7 \end{aligned}$$

otrzymujemy dalsze dwie kongruencje:

$$\log_5 2 + 3 \log_5 3 \equiv 5136 \pmod{10006}$$

oraz

$$3 \log_5 3 + \log_5 7 \equiv 9865 \pmod{10006}.$$

Mamy teraz trzy kongruencje z trzema niewiadomymi. W tym przypadku tak się składa, że istnieje jednoznaczne rozwiązanie modulo 10006, a mianowicie  $\log_5 2 = 6578$ ,  $\log_5 3 = 6190$  i  $\log_5 7 = 1301$ .

Przypuśćmy teraz, że chcemy wyznaczyć  $\log_5 9451$  i wybraliśmy „losowy” wykładnik  $s = 7736$ . Obliczamy

$$9451 \cdot 5^{7736} \pmod{10007} = 8400.$$

Ponieważ liczba  $8400 = 2^4 3^{15} 5^1$  rozkłada się nad zbiorem  $\mathcal{B}$ , otrzymujemy równość

$$\begin{aligned} \log_5 9451 &= 4 \log_5 2 + \log_5 3 + 2 \log_5 5 + \log_5 7 - s \pmod{10006} \\ &= 4 \cdot 6578 + 6190 + 2 \cdot 1 + 1301 - 7736 \pmod{10006} \\ &= 6057. \end{aligned}$$

Możemy sprawdzić, że istotnie  $5^{6057} \equiv 9451 \pmod{10007}$ . □

Przeprowadzono heurystyczne analizy różnych wersji tego algorytmu. Przy rozsądnych założeniach asymptotyczny czas działania w etapie wstępny wynosi  $O\left(e^{(1+o(1))\sqrt{\ln p \ln \ln p}}\right)$ , a do znalezienia jednego logarytmu dyskretnego potrzebny jest czas  $O\left(e^{(1/2+o(1))\sqrt{\ln p \ln \ln p}}\right)$ .

### 5.1.2. Bezpieczeństwo bitów logarytmu dyskretnego

Przyjrzyjmy się teraz kwestii częściowej informacji o logarytmie dyskretnym. W szczególności spróbujmy ustalić, czy obliczenie pojedynczego bitu logarytmu dyskretnego jest łatwe czy trudne. Mówiąc dokładniej, rozważymy problem przedstawiony na rysunku 5.5, który nazwiemy **problemem  $i$ -tego bitu**.

**Dane**  $I = (p, \alpha, \beta, i)$ , gdzie  $p$  jest liczbą pierwszą,  $\alpha \in \mathbb{Z}_p^*$  jest elementem pierwotnym,  $\beta \in \mathbb{Z}_p^*$ ,  $i$  jest liczbą całkowitą, taką że  $1 \leq i \leq \lceil \log_2 p - 1 \rceil$ .

**Cel** Obliczyć  $L_i(\beta)$ , które (dla ustalonych  $\alpha$  i  $p$ ) oznacza  $i$ -ty najmniej znaczący bit  $\log_\alpha \beta$ .

RYSUNEK 5.5.  $i$ -ty bit logarytmu dyskretnego

Na początek stwierdzimy, że obliczenie najmniej znaczącego bitu logarytmu dyskretnego jest łatwe. Innymi słowy, gdy  $i = 1$ , problem ***i-tego bitu*** można efektywnie rozwiązać. Wynika to z kryterium Eulera dla reszt kwadratowych modulo  $p$ , gdzie  $p$  jest liczbą pierwszą.

Rozważmy przekształcenie  $f : \mathbb{Z}_p^* \rightarrow \mathbb{Z}_p^*$  określone w sposób następujący:

$$f(x) = x^2 \pmod{p}.$$

Niech  $\text{QR}(p)$  oznacza zbiór wszystkich reszt kwadratowych modulo  $p$ ; wtedy

$$\text{QR}(p) = \{x^2 \pmod{p} : x \in \mathbb{Z}_p^*\}.$$

Zauważmy, po pierwsze, że  $f(x) = f(p - x)$ . Dalej, że

$$w^2 \equiv x^2 \pmod{p}$$

wtedy i tylko wtedy, gdy

$$p \mid (w - x)(w + x),$$

co z kolei ma miejsce wtedy i tylko wtedy, gdy

$$w \equiv \pm x \pmod{p}.$$

Wynika stąd, że

$$|f^{-1}(y)| = 2$$

dla każdego  $y \in \text{QR}(p)$ , a stąd

$$|\text{QR}(p)| = \frac{p-1}{2}.$$

Tak więc dokładnie połowa reszt w  $\mathbb{Z}_p^*$  to reszty kwadratowe, pozostała połowa składa się z niereszt kwadratowych.

Załóżmy teraz, że  $\alpha$  jest elementem pierwotnym w  $\mathbb{Z}_p$ . Wówczas  $\alpha^a \in \text{QR}(p)$ , gdy  $a$  jest liczbą parzystą. Z tego, że wszystkie elementy  $\alpha^0 \pmod{p}, \alpha^2 \pmod{p}, \dots, \alpha^{p-3} \pmod{p}$  są różne, wnioskujemy, że

$$\text{QR}(p) = \{\alpha^{2i} \pmod{p} : 0 \leq i \leq (p-3)/2\}.$$

Zatem  $\beta$  jest resztą kwadratową wtedy i tylko wtedy, gdy  $\log_{\alpha} \beta$  jest parzysty, a więc wtedy i tylko wtedy, gdy  $L_1(\beta) = 0$ . Wiemy już jednak z kryterium Eulera, że  $\beta$  jest resztą kwadratową wtedy i tylko wtedy, gdy

$$\beta^{(p-1)/2} \equiv 1 \pmod{p}.$$

Mamy zatem następujący efektywny wzór do obliczenia  $L_1(\beta)$ :

$$L_1(\beta) = \begin{cases} 0, & \text{gdy } \beta^{(p-1)/2} \equiv 1 \pmod{p}, \\ 1, & \text{w przeciwnym przypadku.} \end{cases}$$

Rozważmy teraz obliczenia  $L_i(\beta)$ , gdy  $i > 1$ . Niech

$$p - 1 = 2^s t,$$

gdzie  $t$  jest liczbą nieparzystą. Można wówczas wykazać, że  $L_i(\beta)$  daje się łatwo obliczyć, gdy  $i \leq s$ . Z drugiej strony, obliczenie  $L_{s+1}(\beta)$  jest (prawdopodobnie) trudne w tym sensie, że każdy hipotetyczny algorytm (lub wyrocznia) obliczający  $L_{s+1}(\beta)$  mógłby być wykorzystany do znajdowania logarytmów dyskretnych w  $\mathbb{Z}_p$ .

Udowodnimy ten wynik dla  $s = 1$ . Mówiąc dokładniej, jeśli  $p \equiv 3 \pmod{4}$  jest liczbą pierwszą, pokażemy jak można użyć dowolnej wyroczni obliczającej  $L_2(\beta)$  do rozwiązania problemu logarytmu dyskretnego w  $\mathbb{Z}_p$ .

Przypomnijmy, że jeśli  $\beta$  jest resztą kwadratową w  $\mathbb{Z}_p$  oraz  $p \equiv 3 \pmod{4}$ , to pierwiastkami kwadratowymi  $\beta$  modulo  $p$  są liczby  $\pm\beta^{(p+1)/4} \pmod{p}$ . Istotne jest również to, że dla dowolnego  $\beta \neq 0$

$$L_1(\beta) \neq L_1(p - \beta),$$

gdy  $p \equiv 3 \pmod{4}$ . Aby się o tym przekonać, założymy, że

$$\alpha^a \equiv \beta \pmod{p}.$$

Wówczas

$$\alpha^{a+(p-1)/2} \equiv -\beta \pmod{p}.$$

Ponieważ  $p \equiv 3 \pmod{4}$ , zatem liczba całkowita  $(p - 1)/2$  jest nieparzysta, z czego już otrzymujemy oczekiwany wynik.

Niech teraz  $\beta = \alpha^a$  dla pewnego (nieznanego) parzystego wykładnika  $a$ . Wówczas albo

$$\beta^{(p+1)/4} \equiv \alpha^{a/2} \pmod{p},$$

albo

$$-\beta^{(p+1)/4} \equiv \alpha^{a/2} \pmod{p}.$$

Jeśli znamy wartość  $L_2(\beta)$ , to możemy ustalić, która z tych kongruencji ma miejsce, gdyż

$$L_2(\beta) = L_1(\alpha^{a/2}).$$

Ten fakt wykorzystujemy w algorytmie przedstawionym na rysunku 5.6.

Po wykonaniu algorytmu wartości  $x_i$  odpowiadają bitom w reprezentacji binarnej liczby  $\log_\alpha \beta$ , tzn.

$$\log_\alpha \beta = \sum_{i \geq 0} x_i 2^i.$$

Zilustrujmy działanie algorytmu prostym przykładem.

```

(1)  $x_0 = L_1(\beta)$ 
(2)  $\beta = \beta/\alpha^{x_0} \pmod{p}$ 
(3)  $i = 1$ 
(4) while  $\beta \neq 1$  do
(5)    $x_i = L_2(\beta)$ 
(6)    $\gamma = \beta^{(p+1)/4} \pmod{p}$ 
(7)   if  $L_1(\gamma) = x_i$  then
(8)      $\beta = \gamma$ 
(9)   else
(10)     $\beta = p - \gamma$ 
(11)     $\beta = \beta/\alpha^{x_i} \pmod{p}$ 
(12)     $i = i + 1$ 

```

**RYSUNEK 5.6.** Obliczenie logarytmu dyskretnego w  $\mathbb{Z}_p$ , gdy  $p \equiv 3 \pmod{4}$ , przy danej wyroczni dla  $L_2(\beta)$

### Przykład 5.5

Niech  $p = 19$ ,  $\alpha = 2$  oraz  $\beta = 6$ . Przykład jest na tyle mały, że możemy wypisać w tablicy wartości  $L_1(\gamma)$  i  $L_2(\gamma)$  dla wszystkich  $\gamma \in \mathbb{Z}_{19}^*$ . (W ogólności można efektywnie obliczyć  $L_1$  za pomocą kryterium Eulera, a  $L_2$  jest wyrocznią). Podajemy te wartości w tablicy 5.1. Działanie algorytmu przedstawiamy na rysunku 5.7.

Tak więc, jak łatwo sprawdzić,  $\log_2 6 = 1110_2 = 14$ . □

**TABLICA 5.1.** Wartości  $L_1$  i  $L_2$  dla  $p = 19$ ,  $\alpha = 2$

$\gamma$	$L_1(\gamma)$	$L_2(\gamma)$	$\gamma$	$L_1(\gamma)$	$L_2(\gamma)$	$\gamma$	$L_1(\gamma)$	$L_2(\gamma)$
1	0	0	7	0	1	13	1	0
2	1	0	8	1	1	14	1	1
3	1	0	9	0	0	15	1	1
4	0	1	10	1	0	16	0	0
5	0	0	11	0	0	17	0	1
6	0	1	12	1	1	18	1	0

Używając indukcji matematycznej, można podać formalny dowód poprawności algorytmu. Wprowadźmy oznaczenie:

$$x = \log_\alpha \beta = \sum_{i \geq 0} x_i 2^i.$$

Dla  $i \geq 0$  niech

$$Y_i = \left\lfloor \frac{x}{2^{i+1}} \right\rfloor.$$

(1)	$x_0 = 0$
(2)	$\beta = 6$
(3)	$i = 1$
(5)	$x_1 = L_2(6) = 1$
(6)	$\gamma = 5$
(7)	$L_1(5) = 0 \neq x_1$
(10)	$\beta = 14$
(11)	$\beta = 7$
(12)	$i = 2$
(5)	$x_2 = L_2(7) = 1$
(6)	$\gamma = 11$
(7)	$L_1(11) = 0 \neq x_2$
(10)	$\beta = 8$
(11)	$\beta = 4$
(12)	$i = 3$
(5)	$x_3 = L_2(4) = 1$
(6)	$\gamma = 17$
(7)	$L_1(17) = 0 \neq x_3$
(10)	$\beta = 2$
(11)	$\beta = 1$
(12)	$i = 4$
(4)	KONIEC

RYSUNEK 5.7. Obliczenia  $\log_{26}$  w  $\mathbb{Z}_{19}$ 

Zdefiniujmy także  $\beta_0$  jako wartość  $\beta$  w kroku 2 algorytmu oraz, dla  $i \geq 1$ , niech  $\beta_i$  oznacza wartość  $\beta$  w kroku 11 w trakcie  $i$ -tej iteracji pętli **while**. Można dowieść indukcyjnie, że

$$\beta_i \equiv \alpha^{2Y_i} \pmod{p}$$

dla wszystkich  $i \geq 0$ . Dalej, z tego, że

$$2Y_i = Y_{i-1} - x_i$$

wnioskujemy

$$x_{i+1} = L_2(\beta_i)$$

dla  $i \geq 0$ . Wreszcie z równości

$$x_0 = L_1(\beta)$$

wynika poprawność algorytmu. Szczegóły pozostawiamy Czytelnikowi.

## 5.2. Systemy oparte na ciałach skończonych i krzywych eliptycznych

Sporo miejsca poświęciliśmy problemom logarytmu dyskretnego i rozkładu na czynniki pierwsze. Będziemy z nimi jeszcze nieraz się stykać, gdyż leżą one u podłożu różnego rodzaju systemów i protokołów kryptograficznych. Jak do tąd, rozważaliśmy problem logarytmu dyskretnego w skończonym ciele  $\mathbb{Z}_p$ , lecz z podobnym pozytkiem można go rozpatrywać także w innych kontekstach. To właśnie będzie tematem tego podrozdziału.

System kryptograficzny ElGamala można stosować w dowolnej grupie, w której problem logarytmu dyskretnego jest obliczeniowo nieroziągalny. Używaliśmy do tąd grupy modyfikatywnej  $\mathbb{Z}_p^*$ , ale inne grupy też się do tego dobrze nadają. Sformułujemy najpierw problem logarytmu dyskretnego w dowolnej (skończonej) grupie  $G$ , gdzie operację grupową oznaczymy symbolem  $\circ$ . Tę uogólnioną wersję problemu przedstawiamy na rysunku 5.8.

**Dane**  $I = (G, \alpha, \beta)$ , gdzie  $G$  jest skończoną grupą z operacją grupową  $\circ$ ,  $\alpha \in G$  oraz  $\beta \in H$ , przy czym  $H = \{\alpha^i : i \geq 0\}$  jest podgrupą generowaną przez  $\alpha$ .

**Cel** Znaleźć jedyną liczbę całkowitą  $a$ , taką że  $0 \leq a \leq |H| - 1$  oraz  $\alpha^a = \beta$ , gdzie  $\alpha^a$  oznacza

$$\underbrace{\alpha \circ \dots \circ \alpha}_{n \text{ razy}}$$

Taką liczbę  $a$  oznaczymy symbolem  $\log_\alpha \beta$ .

RYSUNEK 5.8. Problem logarytmu dyskretnego w  $(G, \circ)$

Łatwo zdefiniować system kryptograficzny ElGamala w podgrupie  $H$  w sposób podobny do tego, jaki przyjęliśmy pierwotnie w grupie  $\mathbb{Z}_p^*$ . Zrobiliśmy to na rysunku 5.9. Zauważmy, że szyfrowanie wymaga użycia liczby całkowitej  $k$ , takiej że  $0 \leq k \leq |H| - 1$ . Jednakże jeśli Alicja nie zna rzędu podgrupy  $H$ , może wygenerować taką liczbę  $k$ , że  $0 \leq k \leq |G| - 1$ . Wtedy szyfrowanie i deszyfrowanie będzie przebiegało bez żadnych zmian. Zwrócić także uwagę na to, że grupa  $G$  nie musi być abelowa (natomiast abelowa jest, oczywiście, podgrupa  $H$ , ponieważ jest ona podgrupą cykliczną).

Powróćmy do „uogólnionego” problemu logarytmu dyskretnego. Podgrupa  $H$  generowana przez dowolny element  $\alpha \in G$  jest, rzecz jasna, grupą cykliczną rzędu  $|H|$ . Tak więc każda wersja problemu jest w pewnym sensie równoważna problemowi logarytmu dyskretnego w grupie cyklicznej. Wydaje się jednak, że trudność tego problemu zależy w istotny sposób od reprezentacji używanej grupy.

Niech przykładem ukazującym reprezentację, przy której rozwiązanie problemu staje się łatwe, będzie addytywna grupa cykliczna  $\mathbb{Z}_n$  i założymy, że

Niech  $G$  będzie skończoną grupą z operacją grupową  $\circ$  i niech  $\alpha \in G$  będzie elementem takim, że problem logarytmu dyskretnego jest nierozwiązalny w podgrupie  $H = \{\alpha^i : i \geq 0\}$  generowanej przez  $\alpha$ . Przyjmijmy  $\mathcal{P} = G$ ,  $\mathcal{C} = G \times G$ . Określmy

$$\mathcal{K} = \{(G, \alpha, a, \beta) : \beta = \alpha^a\}.$$

Wartości  $\alpha$  i  $\beta$  są znane publicznie, zaś  $a$  jest tajne.

Dla  $K = (G, \alpha, a, \beta)$  i (tajnej) losowej liczby  $k \in \mathbb{Z}_{|H|}$  definiujemy

$$e_K(x, k) = (y_1, y_2),$$

gdzie

$$y_1 = \alpha^k$$

oraz

$$y_2 = x \circ \beta^k.$$

Dla tekstu zaszyfrowanego  $y = (y_1, y_2)$  definiujemy

$$d_K(y) = y_2 \circ (y_1^a)^{-1}.$$

#### RYSUNEK 5.9. Uogólniony system kryptograficzny ElGamala z kluczem publicznym

$\text{NWD}(\alpha, n) = 1$ . Tak więc  $\alpha$  jest generatorem grupy  $\mathbb{Z}_n$ . Operacją grupową jest tu dodawanie modulo  $n$ , zatem operacji „potęgowania”  $\alpha^a$  odpowiada mnożenie przez  $a$  modulo  $n$ . W tym kontekście problem logarytmu dyskretnego sprowadza się do znalezienia liczby całkowitej  $a$ , takiej że

$$\alpha a \equiv \beta \pmod{n}.$$

Ponieważ  $\text{NWD}(\alpha, n) = 1$ , więc istnieje element odwrotny do  $\alpha$  modulo  $n$  i możemy ów element  $\alpha^{-1}$  modulo  $n$  łatwo obliczyć za pomocą algorytmu Euklidesa. Następnie rozwiązujemy kongruencję ze względu na  $a$  i otrzymujemy

$$\log_{\alpha} \beta = \beta \alpha^{-1} \pmod{n}.$$

Omówiliśmy wcześniej problem logarytmu dyskretnego w grupie multiplikatywnej  $\mathbb{Z}_p^*$ , gdzie liczba  $p$  była pierwsza. Jest to grupa cykliczna rzędu  $p-1$ , jest więc izomorficzna z grupą addytywną  $\mathbb{Z}_{p-1}$ . Jak stwierdziliśmy wyżej, umiemy już obliczać logarytmy dyskretne w tej drugiej grupie. Sugeruje to możliwość rozwiązania problemu logarytmu dyskretnego w grupie  $\mathbb{Z}_p^*$  przez sprowadzenie go do łatwego do rozwiązania odpowiednika w grupie  $\mathbb{Z}_{p-1}$ .

Zastanówmy się, jak tego dokonać. Stwierdzenie, że grupa  $(\mathbb{Z}_p^*, \cdot)$  jest izomorficzna z grupą  $(\mathbb{Z}_{p-1}, +)$  oznacza istnienie bijekcji

$$\phi : \mathbb{Z}_p^* \rightarrow \mathbb{Z}_{p-1},$$

takiej że

$$\phi(xy \bmod p) = (\phi(x) + \phi(y)) \bmod (p-1).$$

Wynika stąd natychmiast, że

$$\phi(\alpha^a \bmod p) = a\phi(\alpha) \bmod (p-1),$$

zatem

$$\beta \equiv \alpha^a \pmod{p} \Leftrightarrow a\phi(\alpha) \equiv \phi(\beta) \pmod{p-1}.$$

Obliczając w opisany wyżej sposób niewiadomą  $a$ , otrzymujemy

$$\log_\alpha \beta = \phi(\beta)(\phi(\alpha))^{-1} \bmod (p-1).$$

W rezultacie, jeśli mamy efektywną metodę znajdowania izomorfizmu  $\phi$ , mamy również efektywny algorytm obliczania logarytmów dyskretnych w  $\mathbb{Z}_p^*$ . Trudność polega jednak na tym, że nie znamy żadnej ogólnej metody efektywnego znajdowania izomorfizmu  $\phi$  dla dowolnej liczby pierwszej  $p$ . Nawet gdy wiemy, że dwie grupy są izomorficzne, nie mamy efektywnego algorytmu, który pozwoliłby taki izomorfizm opisać w sposób jawny.

Tę metodę można zastosować do problemu logarytmu dyskretnego w dowolnej grupie  $G$ . Jeśli istnieje efektywna metoda znajdowania izomorfizmu między  $H$  i  $\mathbb{Z}_{|H|}$ , to opisany wyżej problem logarytmu dyskretnego w  $G$  można efektywnie rozwiązać. I na odwrót, nietrudno zauważyc, że efektywna metoda obliczania logarytmów dyskretnych prowadzi do efektywnego algorytmu znajdowania izomorfizmu między tymi dwiema grupami.

Powыższe rozumowanie pokazuje, że problem logarytmu dyskretnego może być łatwy lub (na pierwszy rzut oka) trudny, zależnie od reprezentacji używanej grupy (cyklicznej). Warto więc przyjrzeć się innym grupom w nadziei znalezienia takich, w których problem logarytmu dyskretnego wydaje się być praktycznie nierozwiązalny.

Oto dwie klasy takich grup:

- (1) modyfikatywna grupa ciała Galois  $GF(p^n)$ ;
- (2) grupa krzywej eliptycznej nad ciałem skończonym.

Obie te klasy omówimy w następnych podrozdziałach.

### 5.2.1. Ciało Galois

Stwierdziliśmy wcześniej, że  $\mathbb{Z}_p$  jest ciałem, gdy  $p$  jest liczbą pierwszą. Istnieją jednak inne ciała skończone. W szczególności dla każdej liczby  $q$ , takiej że  $q = p^n$  dla pewnej liczby pierwszej  $p$  i liczby naturalnej  $n \geq 1$ , istnieje ciało  $q$ -elementowe. Opiszymy teraz krótko, jak takie ciało zbudować. Potrzebujemy do tego kilku definicji.

#### DEFINICJA 5.1

Niech  $p$  będzie liczbą pierwszą i niech  $\mathbb{Z}_p[x]$  oznacza zbiór wszystkich wielomianów jednej zmiennej  $x$ . Definiując w tym zbiorze dodawanie i mnożenie wielomianów w zwykły sposób (i redukując współczynniki modulo  $p$ ), otrzymujemy pierścienie.

Jeśli  $f(x), g(x) \in \mathbb{Z}_p[x]$ , to mówimy, że  $f(x)$  dzieli  $g(x)$  (i piszemy  $f(x)|g(x)$ ), gdy istnieje wielomian  $q(x) \in \mathbb{Z}_p[x]$ , taki że

$$g(x) = q(x)f(x).$$

Dla  $f(x) \in \mathbb{Z}_p[x]$  definiujemy  $\deg(f)$ , stopień wielomianu  $f$ , jako najwyższy wykładnik potęgi zmiennej  $x$  w  $f$ .

Niech  $f(x), g(x), h(x) \in \mathbb{Z}_p[x]$  i  $\deg(f) = n \geq 1$ . Przyjmujemy, że

$$g(x) \equiv h(x) \pmod{f(x)},$$

gdy

$$f(x) \mid (g(x) - h(x)).$$

Zwrócić uwagę na podobieństwo definicji kongruencji dla wielomianów do definicji kongruencji dla liczb całkowitych.

Zdefiniujemy teraz pierścień wielomianów „modulo  $f(x)$ ”, który oznaczymy symbolem  $\mathbb{Z}_p[x]/(f(x))$ . Konstrukcja  $\mathbb{Z}_p[x]/(f(x))$ , wychodząca od pierścienia  $\mathbb{Z}_p[x]$ , opiera się na pojęciu kongruencji modulo  $f(x)$  i jest analogiczna do konstrukcji  $\mathbb{Z}_m$  z  $\mathbb{Z}$ .

Niech  $\deg(f) = n$ . Jeśli podzielimy  $g(x)$  przez  $f(x)$ , to otrzymamy (jednoznacznie wyznaczony) iloraz  $q(x)$  oraz resztę  $r(x)$ , gdzie

$$g(x) = q(x)f(x) + r(x)$$

oraz

$$\deg(r) < n.$$

Iloraz i resztę można uzyskać drogą zwykłego dzielenia wielomianów. Stąd każdy wielomian w  $\mathbb{Z}_p[x]$  przystaje modulo  $f(x)$  do pewnego wielomianu stopnia co najwyżej  $n - 1$ .

Każdy element  $\mathbb{Z}_p[x]/(f(x))$  możemy teraz utożsamić z jednym z  $p^n$  wielomianów stopnia co najwyżej  $n - 1$ . Dodawanie i mnożenie jest określone

w  $\mathbb{Z}_p[x]/(f(x))$  niemal tak jak w  $\mathbb{Z}_p[x]$ , z tym że wynik działania należy jeszcze zredukować modulo  $f(x)$ . Zbiór  $\mathbb{Z}_p[x]/(f(x))$  z tymi operacjami staje się pierścieniem.

Przypomnijmy, że  $\mathbb{Z}_m$  jest pierścieniem wtedy i tylko wtedy, gdy  $m$  jest liczbą pierwszą, a odwrotności elementów tego pierścienia można obliczyć za pomocą algorytmu Euklidesa. Podobnie jest w  $\mathbb{Z}_p[x]/(f(x))$ . Odpowiednikiem pierwszości jest tu pojęcie nierozkładalności, które zdefiniujemy w sposób następujący:

### DEFINICJA 5.2

Wielomian  $f(x) \in \mathbb{Z}_p[x]$  nazwiemy *nierozkładalnym*, jeśli nie istnieją wielomiany  $f_1(x), f_2(x) \in \mathbb{Z}_p[x]$ , takie że

$$f(x) = f_1(x)f_2(x),$$

gdzie  $\deg(f_1) > 0$  oraz  $\deg(f_2) > 0$ .

Bardzo istotną właściwością pierścienia  $\mathbb{Z}_p[x]/(f(x))$  jest to, że jest on ciałem wtedy i tylko wtedy, gdy wielomian  $f(x)$  jest nierozkładalny. Ponadto, odwrotności elementów tego pierścienia można obliczyć za pomocą prostej modyfikacji (rozszerzonego) algorytmu Euklidesa.

Oto przykład ilustrujący wprowadzone pojęcia.

### Przykład 5.6

Spróbujmy zbudować ciało 8-elementowe. Można to zrobić, znajdująając nierozkładalny wielomian trzeciego stopnia w  $\mathbb{Z}_2[x]$ . Wystarczy w tym celu rozważyć tylko wielomiany z wyrazem stałym równym 1, gdyż każdy wielomian z zerowym wyrazem stałym jest podzielny przez  $x$ , a więc rozkładalny. Istnieją cztery takie wielomiany:

$$f_1(x) = x^3 + 1$$

$$f_2(x) = x^3 + x + 1$$

$$f_3(x) = x^3 + x^2 + 1$$

$$f_4(x) = x^3 + x^2 + x + 1.$$

Wielomian  $f_1(x)$  jest rozkładalny, gdyż

$$x^3 + 1 = (x + 1)(x^2 + x + 1)$$

(pamiętajmy, że wszystkie współczynniki są zredukowane modulo 2). Również wielomian  $f_4(x)$  jest rozkładalny, gdyż

$$x^3 + x^2 + x + 1 = (x + 1)(x^2 + 1).$$

Jednakże oba pozostałe wielomiany,  $f_2(x)$  i  $f_3(x)$ , są nierozkładalne i każdy z nich może posłużyć do budowy ciała o ósmiu elementach.

Użyjmy  $f_2(x)$  i zbudujmy ciało  $\mathbb{Z}_2[x]/(x^3 + x + 1)$ . Ciało to składa się z ósmiu wielomianów: 0, 1,  $x$ ,  $x + 1$ ,  $x^2$ ,  $x^2 + 1$ ,  $x^2 + x$  oraz  $x^2 + x + 1$ .

Aby obliczyć iloczyn dwóch elementów tego ciała, mnożymy dwa wielomiany i redukujemy wynik modulo  $x^3 + x + 1$  (czyli dzielimy go przez  $x^3 + x + 1$  i bierzemy resztę). Ponieważ dzielimy przez wielomian trzeciego stopnia, reszta ma stopień co najwyżej drugi, jest więc elementem ciała.

Obliczmy, na przykład, iloczyn  $(x^2 + 1)(x^2 + x + 1)$  w  $\mathbb{Z}_2[x]/(x^3 + x + 1)$ . Najpierw obliczamy ten iloczyn w  $\mathbb{Z}_2[x]$  i otrzymujemy  $x^4 + x^3 + x + 1$ . Teraz dzielimy przez  $x^3 + x + 1$  i dochodzimy do równości

$$x^4 + x^3 + x + 1 = (x + 1)(x^3 + x + 1) + x^2 + x.$$

Tak więc w ciele  $\mathbb{Z}_2[x]/(x^3 + x + 1)$  mamy

$$(x^2 + 1)(x^2 + x + 1) = x^2 + x.$$

Poniżej przedstawiamy pełną tabliczkę mnożenia dla niezerowych elementów ciała. Dla oszczędności miejsca zapisujemy wielomian postaci  $a_2x^2 + a_1x + a_0$  jako trójkę  $a_2a_1a_0$ .

	001	010	011	100	101	110	111
001	001	010	011	100	101	110	111
010	010	100	110	011	001	111	101
011	011	110	101	111	100	001	010
100	100	011	111	110	010	101	001
101	101	001	100	010	111	011	110
110	110	111	001	101	011	010	100
111	111	101	010	001	110	100	011

Do obliczania elementów odwrotnych można użyć bezpośredniej adaptacji rozszerzonego algorytmu Euklidesa.

Zauważmy na koniec, że grupa multiplikatywna niezerowych wielomianów nad rozważanym ciałem jest grupą cykliczną rzędu 7. Liczba 7 jest pierwsza, więc każdy niezerowy element ciała jest generatorem tej grupy, czyli elementem pierwotnym ciała.

Obliczając, na przykład, potęgi elementu  $x$ , otrzymujemy:

$$x^1 = x$$

$$x^2 = x^2$$

$$x^3 = x + 1$$

$$x^4 = x^2 + x$$

$$x^5 = x^2 + x + 1$$

$$x^6 = x^2 + 1$$

$$x^7 = 1,$$

a więc wszystkie niezerowe elementy ciała. □

Pozostaje rozważyć jeszcze pytanie o istnienie i jednoznaczność ciał tego typu. Można wykazać, że w  $\mathbb{Z}_p[x]$  istnieje co najmniej jeden wielomian nierozkładalny

danego stopnia  $n \geq 1$ . Stąd wynika istnienie skończonego ciała o  $p^n$  elementach, dla każdej liczby pierwszej  $p$  i dowolnej liczby naturalnej  $n \geq 1$ . Zwykle w  $\mathbb{Z}_p[x]$  istnieje wiele wielomianów nieroziędalnych stopnia  $n$ , można jednak udowodnić, że skończone ciała zbudowane na dowolnych dwóch wielomianach nieroziędalnych stopnia  $n$  są izomorficzne. Istnieje zatem jedyne skończone ciało o liczbie elementów równej  $p^n$ , dla dowolnej liczby pierwszej  $p$  i  $n \geq 1$ . Oznaczmy je symbolem  $\text{GF}(p^n)$ . W przypadku gdy  $n = 1$ , otrzymane w ten sposób ciało  $\text{GF}(p^n)$  jest po prostu tym samym, co  $\mathbb{Z}_p$ . Można wreszcie wykazać, że jeśli  $r$  nie jest liczbą postaci  $p^n$  dla pewnej liczby pierwszej  $p$  i liczby naturalnej  $n \geq 1$ , to nie istnieje skończone ciało  $r$ -elementowe.

Zauważliśmy już, że grupa multiplikatywna  $\mathbb{Z}_p^*$  (dla liczby pierwszej  $p$ ) jest grupą cykliczną rzędu  $p - 1$ . W rzeczywistości każda grupa multiplikatywna ciała skończonego jest cykliczna:  $\text{GF}(p^n) \setminus \{0\}$  jest grupą cykliczną rzędu  $p^n - 1$ . Dostajemy w ten sposób dalsze przykłady grup cyklicznych, w których można rozważyć problem logarytmu dyskretnego.

W praktyce najlepiej są zbadane ciała skończone  $\text{GF}(2^n)$ . Algorytmy dla problemu logarytmu dyskretnego zarówno Shanksa, jak i Pohliga–Hellmana, działają w ciałach  $\text{GF}(2^n)$ . Z kolei metodę obliczania indeksu można dostosować tak, by była także skuteczna w takich ciałach. Czas obliczeń wstępnych algorytmu obliczania indeksu okazuje się równy

$$O\left(e^{(1,405+o(1))n^{1/3}(\ln n)^{2/3}}\right),$$

podczas gdy czas potrzebny do obliczenia jednego logarytmu dyskretnego wynosi

$$O\left(e^{(1,098+o(1))n^{1/3}(\ln n)^{2/3}}\right).$$

Uważa się jednak, że dla dużych wartości  $n$  (powiedzmy  $n > 800$ ) problem logarytmu dyskretnego w  $\text{GF}(2^n)$  jest praktycznie nieroziędalny, jeśli tylko liczba  $2^n - 1$  ma co najmniej jeden „duży” czynnik pierwszy (potrzebny do powstrzymania ataku z użyciem algorytmu Pohliga–Hellmana).

### 5.2.2. Krzywe eliptyczne

Zaczniemy od definicji krzywej eliptycznej.

#### DEFINICJA 5.3

Niech  $p > 3$  będzie liczbą pierwszą. Krzywą eliptyczną  $y^2 = x^3 + ax + b$  nad  $\mathbb{Z}_p$  jest zbiór rozwiązań  $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$  kongruencji

$$y^2 \equiv x^3 + ax + b \pmod{p}, \tag{5.1}$$

gdzie  $a, b \in \mathbb{Z}_p$  są stałymi, takimi że  $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$ , wraz z wyróżnionym punktem  $\mathcal{O}$ , zwany punktem w nieskończoności<sup>1</sup>.

<sup>1</sup>Równanie (5.1) może posłużyć do zdefiniowania krzywej eliptycznej nad dowolnym ciałem  $\text{GF}(p^n)$  dla liczby pierwszej  $p > 3$ . Krzywą eliptyczną nad  $\text{GF}(2^n)$  lub  $\text{GF}(3^n)$  definiuje się nieco innym równaniem.

Krzywą eliptyczną  $E$  można przekształcić w grupę abelową, określając odpowiednią operację na jej punktach. Operację zapisuje się zazwyczaj addytywnie i określa w następujący sposób (wszystkie działania arytmetyczne wykonywane są w  $\mathbb{Z}_p$ ): niech

$$P = (x_1, y_1)$$

oraz

$$Q = (x_2, y_2)$$

będą punktami na krzywej  $E$ . Jeśli  $x_2 = x_1$  i  $y_2 = -y_1$ , to  $P + Q = \mathcal{O}$ ; w przeciwnym razie  $P + Q = (x_3, y_3)$ , gdzie

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda(x_1 - x_3) - y_1$$

oraz

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & \text{gdy } P \neq Q; \\ \frac{3x_1^2 + a}{2y_1}, & \text{gdy } P = Q. \end{cases}$$

Określmy wreszcie

$$P + \mathcal{O} = \mathcal{O} + P = P$$

dla wszystkich  $P \in E$ . Można wykazać, że  $E$  z taką operacją dodawania jest grupą abelową, w której elementem neutralnym względem dodawania jest  $\mathcal{O}$  (sprawdzenie większości własności jest łatwe, choć żmudne; jedynie dowód łączności jest dość trudny).

Zauważmy, że elementy przeciwnie oblicza się w tej grupie bardzo łatwo. Elementem przeciwnym do elementu  $(x, y)$  (oznaczymy go jako  $-(x, y)$ , gdyż grupa jest addytywna) jest punkt  $(x, -y)$ , dla dowolnego punktu  $(x, y) \in E$ .

Popatrzmy na prosty przykład.

### Przykład 5.7

Niech  $E$  będzie krzywą eliptyczną  $y^2 = x^3 + x + 6$  nad  $\mathbb{Z}_{11}$ . Określmy najpierw punkty należące do  $E$ . Można w tym celu sprawdzić wszystkie elementy  $x \in \mathbb{Z}_{11}$ , obliczając  $x^3 + x + 6 \bmod 11$ , a następnie starając się rozwiązać równanie (5.1) ze względu na  $y$ . Dla danego  $x$  sprawdzamy za pomocą kryterium Eulera, czy  $z = x^3 + x + 6 \bmod 11$  jest resztą kwadratową. Przypomnijmy, że istnieje jawnny wzór umożliwiający obliczenie pierwiastków kwadratowych z resztą kwadratowymi modulo  $p$ , dla liczb pierwszych  $p \equiv 3 \pmod{4}$ . Stosując ten wzór, otrzymujemy następujące pierwiastki kwadratowe z resztą kwadratową  $z$ :

$$\pm z^{(11+1)/4} \bmod 11 \equiv \pm z^3 \bmod 11.$$

Wyniki tych obliczeń zawiera tablica 5.2.

**TABLICA 5.2. Punkty krzywej eliptycznej  $y^2 = x^3 + x + 6$  nad  $\mathbb{Z}_{11}$**

$x$	$x^3 + x + 6 \bmod 11$	Jest w $\text{QR}(11)$ ?	$y$
0	6	nie	
1	8	nie	
2	5	tak	4, 7
3	3	tak	5, 6
4	8	nie	
5	4	tak	2, 9
6	8	nie	
7	4	tak	2, 9
8	9	tak	3, 8
9	7	nie	
10	4	tak	2, 9

Tak więc do  $E$  należy 13 punktów. Ponieważ każda grupa, której rząd jest liczbą pierwszą jest cykliczna, zatem  $E$  jest izomorficzna z  $\mathbb{Z}_{13}$  i każdy punkt różny od punktu w nieskończoności jest jej generatorem. Weźmy generator  $\alpha = (2, 7)$ . Możemy wówczas obliczyć „potęgi”  $\alpha$  (zapiszemy je jako wielokrotności  $\alpha$ , gdyż operacją grupową jest dodawanie). Obliczenie  $2\alpha = (2, 7) + (2, 7)$  zaczynamy od obliczenia  $\lambda$ :

$$\begin{aligned}\lambda &= (3 \cdot 2^2 + 1)(2 \cdot 7)^{-1} \bmod 11 \\ &= 2 \cdot 3^{-1} \bmod 11 \\ &= 2 \cdot 4 \bmod 11 \\ &= 8.\end{aligned}$$

Mamy wówczas

$$\begin{aligned}x_3 &= 8^2 - 2 - 2 \bmod 11 \\ &= 5\end{aligned}$$

oraz

$$\begin{aligned}y_3 &= 8(2 - 5) - 7 \bmod 11 \\ &= 2,\end{aligned}$$

a zatem  $2\alpha = (5, 2)$ .

Kolejną wielokrotnością jest  $3\alpha = 2\alpha + \alpha = (5, 2) + (2, 7)$ . I znów zaczynamy od obliczenia  $\lambda$ , co w tej sytuacji wygląda tak:

$$\begin{aligned}\lambda &= (7 - 2)(2 - 5)^{-1} \bmod 11 \\ &= 5 \cdot 8^{-1} \bmod 11 \\ &= 5 \cdot 7 \bmod 11 \\ &= 2.\end{aligned}$$

Mamy wówczas

$$\begin{aligned}x_3 &= 2^2 - 5 - 2 \bmod 11 \\&= 8\end{aligned}$$

oraz

$$\begin{aligned}y_3 &= 2(5 - 8) - 2 \bmod 11 \\&= 3,\end{aligned}$$

a zatem  $3\alpha = (8, 3)$ .

Postępując tak dalej, otrzymujemy wszystkie wielokrotności  $\alpha$ :

$$\begin{array}{lll}\alpha = (2, 7) & 2\alpha = (5, 2) & 3\alpha = (8, 3) \\4\alpha = (10, 2) & 5\alpha = (3, 6) & 6\alpha = (7, 9) \\7\alpha = (7, 2) & 8\alpha = (3, 5) & 9\alpha = (10, 9) \\10\alpha = (8, 8) & 11\alpha = (5, 9) & 12\alpha = (2, 4)\end{array}$$

Tak więc  $\alpha = (2, 7)$  jest rzeczywiście elementem pierwotnym. □

Liczba punktów należących do krzywej eliptycznej  $E$  określonej nad ciałem  $\mathbb{Z}_p$  ( $p$  jest liczbą pierwszą,  $p > 3$ ) jest zbliżona do  $p$ . Dokładniej, znane twierdzenie Hassego orzeka, iż liczba punktów krzywej eliptycznej  $E$ ,  $\#E$ , spełnia nierówności

$$p + 1 - 2\sqrt{p} \leq \#E \leq p + 2\sqrt{p}.$$

Obliczenie dokładnej wartości  $\#E$  jest trudniejsze, istnieje jednak opracowany przez Schoofa efektywny algorytm wykonujący takie obliczenia. (Termin „efektywny” oznacza tu, że czas działania algorytmu jest wielomianowy ze względu na  $\log p$ . Czas działania algorytmu Schoofa wynosi  $O((\log p)^8)$  operacji bitowych, dzięki czemu jest on praktycznie stosowalny dla kilkusetcyfrowych liczb pierwszych  $p$ ).

Zakładając, że umiemy obliczyć  $\#E$ , chcemy dalej znaleźć podgrupę cykliczną grupy  $E$ , w której problem logarytmu dyskretnego byłby praktycznie nieroziwiązalny. Chcielibyśmy zatem coś wiedzieć o strukturze grupy  $E$ . Następujące twierdzenie dostarcza o tej strukturze istotnej informacji.

### **TWIERDZENIE 5.1**

Niech  $E$  będzie krzywą eliptyczną określoną nad  $\mathbb{Z}_p$ , gdzie  $p$  jest liczbą pierwszą i  $p > 3$ . Wówczas istnieją liczby całkowite  $n_1$  i  $n_2$ , takie że  $E$  jest izomorficzna z  $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$ . Ponadto,  $n_2|n_1$  i  $n_2|(p-1)$ .

Wynika stąd, że jeśli potrafimy znaleźć liczby  $n_1$  i  $n_2$ , to otrzymujemy podgrupę cykliczną grupy  $E$ , izomorficzną z grupą  $\mathbb{Z}_{n_1}$ , która może potencjalnie służyć za podstawę systemu kryptograficznego ElGamala.

Zauważmy, że jeśli  $n_2 = 1$ , to  $E$  jest grupą cykliczną. Podobnie,  $E$  jest grupą cykliczną, gdy  $\#E$  jest liczbą pierwszą lub iloczynem różnych liczb pierwszych.

W algorytmach Shanksa i Pohliga–Hellmana stosuje się problem logarytmu dyskretnego w krzywych eliptycznych, natomiast nie jest znana żadna adaptacja metody obliczania indeksu do tych krzywych. Istnieje jednak metoda wykorzystania jawnego izomorfizmu między krzywymi eliptycznymi i ciałami skończonymi, prowadząca do efektywnych algorytmów dla pewnych klas krzywych. Można tę technikę, zaproponowaną przez Menezesa, Okamoto i Vanstone'a, stosować w pewnych przypadkach w klasie szczególnych krzywych eliptycznych, zwanych krzywymi supersobliwymi, które znalazły się w kręgu zainteresowań kryptografów. Jeśli jednak chce się unikać krzywych supersobliwych, to można przyjąć, że krzywa eliptyczna z podgrupą cykliczną rzędu około  $2^{160}$  zapewnia, jak się wydaje, bezpieczeństwo systemu kryptograficznego, pod warunkiem że rząd podgrupy jest podzielny przez co najmniej jeden duży czynnik pierwszy (jak poprzednio w celu obrony przed atakiem Pohliga–Hellmana).

Popatrzmy teraz na przykład szyfrowania w systemie ElGamala z użyciem krzywej eliptycznej z przykładu 5.7.

### Przykład 5.8

Załóżmy, że  $\alpha = (2, 7)$ , a tajnym „wykładnikiem” Bolka jest  $a = 7$ . Wówczas

$$\beta = 7\alpha = (7, 2).$$

Operacja szyfrowania ma więc postać następującą:

$$e_K(x, k) = (k(2, 7), x + k(7, 2)),$$

gdzie  $x \in E$  oraz  $0 \leq k \leq 12$ , podczas gdy operacją deszyfrowania jest

$$d_K(y_1, y_2) = y_2 - 7y_1.$$

Przypuśćmy, że Alicja chce zaszyfrować komunikat  $x = (10, 9)$  (punkt na krzywej  $E$ ). Jeśli wybierze losową wartość  $k = 3$ , wykona następujące obliczenia:

$$\begin{aligned} y + 1 &= 3(2, 7) \\ &= (8, 3) \end{aligned}$$

oraz

$$\begin{aligned} y_2 &= (10, 9) + 3(7, 2) \\ &= (10, 9) + (3, 5) \\ &= (10, 2). \end{aligned}$$

W rezultacie  $y = ((8, 3), (10, 2))$ . Gdy teraz Bolek dostanie tekst zaszyfrowany  $y$ , odczyta go w sposób następujący:

$$\begin{aligned} x &= (10, 2) - 7(8, 3) \\ &= (10, 2) - (3, 5) \\ &= (10, 2) + (3, 6) \\ &= (10, 9). \end{aligned}$$

Rezultat jest więc zgodny z pierwotnym tekstem jawnym.  $\square$

Przy implementacji systemu kryptograficznego ElGamala na krzywych eliptycznych pojawiają się pewne trudności praktyczne. Gdy stosuje się ten system w  $\mathbb{Z}_p$  (lub w  $GF(p^n)$  dla  $n > 1$ ), można dwukrotnie zwiększyć wielkość komunikatu, natomiast w przypadku implementacji na krzywej eliptycznej można zwiększyć go mniej więcej czterokrotnie. Bierze się to stąd, że mamy  $p$  tekstów jawnych, ale każdy tekst zaszyfrowany składa się z czterech elementów ciała. Poważniejszym utrudnieniem jest jednak to, że przestrzeń tekstów jawnych składa się z punktów na krzywej  $E$ , a nie jest znana żadna metoda deterministycznego generowania tych punktów.

Menezes i Vanstone zaproponowali wariant bardziej efektywny. Krzywej eliptycznej używa się w nim do „maskowania”, natomiast tekstami jawnymi lub zaszyfrowanymi mogą być dowolne pary uporządkowane (niezerowych) elementów ciała (a więc nie muszą to być punkty krzywej  $E$ ). Daje to współczynnik powiększenia komunikatu równy dwa, podobnie jak w oryginalnym systemie ElGamala. System kryptograficzny **Menezesa–Vanstone'a** jest pokazany na rysunku 5.10.

Powracając do krzywej  $y^2 = x^3 + x + 6$  nad  $\mathbb{Z}_{11}$ , widzimy, że system Menezesa–Vanstone'a dopuszcza  $10 \cdot 10 = 100$  tekstów jawnych, podczas gdy oryginalny system jedynie 13. Użyjemy tej samej krzywej do ilustracji procesu szyfrowania i deszyfrowania w systemie Menezesa–Vanstone'a.

### Przykład 5.9

Jak w poprzednim przykładzie niech  $\alpha = (2, 7)$  i niech tajnym „wykładownikiem” Bolka będzie  $a = 7$ ; stąd

$$\beta = 7\alpha = (7, 2).$$

Przypuśćmy, że Alicja chce zaszyfrować tekst jawny

$$x = (x_1, x_2) = (9, 1)$$

(zwróćmy uwagę, że  $x$  nie jest punktem krzywej  $E$ ) i wybiera wartość losową  $k = 6$ . Najpierw oblicza

$$y_0 = k\alpha = 6(2, 7) = (7, 9)$$

Niech  $E$  będzie krzywą eliptyczną określoną na  $\mathbb{Z}_p$  ( $p > 3$ , liczba pierwsza), taką że  $E$  zawiera cykliczną podgrupę  $H$ , w której problem logarytmu dyskretnego jest obliczeniowo nieroziągalny.

Przyjmijmy  $\mathcal{P} = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ ,  $\mathcal{C} = E \times \mathbb{Z}_p^* \times \mathbb{Z}_p^*$  i określmy

$$\mathcal{K} = \{(E, \alpha, a, \beta) : \beta = a\alpha\},$$

gdzie  $\alpha \in E$ . Wartości  $\alpha$  i  $\beta$  są znane publicznie, liczba  $a$  jest tajna.

Dla  $K = (E, \alpha, a, \beta)$ , (tajnej) losowej liczby  $k \in \mathbb{Z}_{|H|}$  oraz  $x = (x_1, x_2) \in \mathbb{Z}_p^* \times \mathbb{Z}_p^*$  definiujemy

$$e_K(x, k) = (y_0, y_1, y_2),$$

gdzie

$$y_0 = k\alpha$$

$$(c_1, c_2) = k\beta$$

$$y_1 = c_1 x_1 \bmod p$$

$$y_2 = c_2 x_2 \bmod p.$$

Dla tekstu zaszyfrowanego  $y = (y_0, y_1, y_2)$  definiujemy

$$d_K(y) = (y_1 c_1^{-1} \bmod p, y_2 c_2^{-1} \bmod p),$$

gdzie

$$ay_0 = (c_1, c_2).$$

#### RYSUNEK 5.10. Kryptosystem Menezesa–Vanstone'a oparty na krzywej eliptycznej

oraz

$$k\beta = 6(7, 2) = (8, 3),$$

a zatem  $c_1 = 8$  i  $c_2 = 3$ .

Dalej oblicza

$$y_1 = c_1 x_1 \bmod 11 = 8 \cdot 9 \bmod 11 = 6$$

oraz

$$y_2 = c_2 x_2 \bmod 11 = 3 \cdot 1 \bmod 11 = 3.$$

Tekstem zaszyfrowanym, który Alicja wyśle Bolkowi, jest

$$y = (y_0, y_1, y_2) = ((7, 9), 6, 3).$$

Gdy Bolek dostanie tekst  $y$ , obliczy najpierw

$$(c_1, c_2) = ay_0 = 7(7, 9) = (8, 3),$$

a następnie

$$\begin{aligned} x &= (y_1 c_1^{-1} \bmod p, y_2 c_2^{-1} \bmod p) \\ &= (6 \cdot 8^{-1} \bmod 11, 3 \cdot 3^{-1} \bmod 11) \\ &= (6 \cdot 7 \bmod 11, 3 \cdot 4 \bmod 11) \\ &= (9, 1). \end{aligned}$$

Proces deszyfrowania dał więc poprawny wynik. □

### 5.3. System plecakowy Merkle'a–Hellmana

Dobrze znany system kryptograficzny Merkle'a–Hellmana został przez obu autorów po raz pierwszy opisany w 1978 roku. Mimo iż został on, a także kilka jego wariantów, złamany we wczesnych latach osiemdziesiątych ubiegłego wieku, warto mu się przyjrzeć z badawczą uwagą ze względu na jego elegancję pojęciową i technikę projektową.

W gruncie rzeczy nazwa „plecakowy” nie jest tu odpowiednia<sup>2</sup>; system opiera się na problemie sumy podzbioru, opisany na rysunku 5.11.

**Dane**  $I = (s_1, \dots, s_n, T)$ , gdzie  $s_1, \dots, s_n$  i  $T$  są dodatnimi liczbami całkowitymi. Liczby  $s_i$  nazywamy *wagami*, liczba  $T$  jest *sumą docelową*.

**Pytanie** Czy istnieje zero-jedynkowy wektor  $\mathbf{x} = (x_1, \dots, x_n)$ , taki że

$$\sum_{i=1}^n x_i s_i = T?$$

RYSUNEK 5.11. Problem sumy podzbioru

**Problem sumy podzbioru**, tak jak go przedstawiliśmy na rysunku 5.11, jest problemem *decyzyjnym* (odpowiedzią jest „tak” lub „nie”). Jeśli go nieco przeformułujemy, tak aby w przypadku, gdy odpowiedź brzmi „tak”, otrzymać

<sup>2</sup> Problem **plecakowy**, tak jak go się zwykle definiuje, polega na wyborze obiektów o przypisanych im wagach i zyskach, tak by nie przekroczyć wskazanej objętości i uzyskać przewidywany zysk.

odpowiedni wektor  $\mathbf{x}$  (niekoniecznie jedyny), to będziemy mieli do czynienia z problemem *wyszukiwania*.

Problem (decyzyjny) sumy podzbioru jest jednym z problemów określanych jako NP-zupełne. Oznacza to, między innymi, że nie istnieje rozwiążający go algorytm działający w czasie wielomianowym. Tak jest również dla problemu wyszukiwania sumy podzbioru. Jednakże jeśli nawet nie istnieje działający w czasie wielomianowym algorytm dla przypadku ogólnego, to nie wyklucza to istnienia takiego algorytmu dla niektórych szczególnych przypadków. Tak właśnie jest z problemem sumy podzbioru.

Mówimy, że ciąg wag  $(s_1, \dots, s_n)$  jest *superrosnący*, gdy

$$s_j > \sum_{i=1}^{j-1} s_i$$

dla  $2 \leq j \leq n$ . Gdy ciąg wag jest superrosnący, problem wyszukiwania sumy podzbioru może być łatwo rozwiązany w czasie  $O(n)$ , a otrzymane rozwiązanie  $\mathbf{x}$  (jeśli istnieje) musi być jedyne. Daje je algorytm przedstawiony na rysunku 5.12.

```

(1) for  $i = n$  downto 1 do
(2)   if  $T \geq s_i$  then
(3)      $T = T - s_i$ 
(4)      $x_i = 1$ 
(5)   else
(6)      $x_i = 0$ 
(7) if  $T = 0$  then
(8)    $X = (x_1, \dots, x_n)$  jest rozwiązaniem
(9) else
(10)  rozwiązania nie ma

```

RYSUNEK 5.12. Algorytm rozwiążający problem sumy podzbioru dla ciągu superroszącego

Niech  $\mathbf{s} = (s_1, \dots, s_n)$  będzie ciągiem superrosącym. Rozważmy funkcję

$$e_{\mathbf{s}} : \{0, 1\}^n \rightarrow \left\{ 0, \dots, \sum_{i=1}^n s_i \right\}$$

określoną wzorem

$$e_{\mathbf{s}}(x_1, \dots, x_n) = \sum_{i=1}^n x_i s_i.$$

Czy  $e_{\mathbf{s}}$  jest dobrym kandydatem na regułę szyfrowania? Z tego, że ciąg  $\mathbf{s}$  jest superroszący, wynika, że  $e_{\mathbf{s}}$  jest funkcją różnowartościową, a wtedy algorytm

przedstawiony na rysunku 5.12 byłby odpowiadającym jej algorytmem deszyfrowania. Taki system nie dawałby jednak żadnej gwarancji bezpieczeństwa, gdyż ktokolwiek (łącznie z Oskarem) mógłby odczytać komunikat zaszyfrowany w taki sposób.

Strategia musi więc uwzględnić konieczność takiego przekształcenia ciągu wag, by przestał on być superrosnący. Aby odzyskać ciąg superrosnący, Bolek zastosuje przekształcenie odwrotne. Natomiast Oskar, przystępując do łamania szyfru i nie wiedząc, jakie przekształcenie zostało zastosowane, zobaczy jedynie ogólny, zapewne trudny przypadek problemu sumy podzbioru.

Jeden z odpowiednich typów przekształceń stanowią przekształcenia *modularne*. Wybieramy jako moduł liczbę pierwszą  $p$ , taką że

$$p > \sum_{i=1}^n s_i,$$

oraz mnożnik  $a$ , taki że  $1 \leq a \leq p - 1$ . Teraz definiujemy

$$t_i = a s_i \bmod p$$

dla  $1 \leq i \leq n$ . Kluczem publicznym szyfrowania będzie właśnie ciąg wag  $\mathbf{t} = (t_1, \dots, t_n)$ . Wartości  $a$  i  $p$ , użyte do określenia przekształcenia modułarnego, pozostają tajne. Pełny opis systemu plecakowego Merkle'a–Hellmana znajduje się na rysunku 5.13.

Zobaczmy, jak przebiegają procesy szyfrowania i deszyfrowania w systemie Merkle'a–Hellmana na prostym przykładzie.

### Przykład 5.10

Niech

$$\mathbf{s} = (2, 5, 9, 21, 45, 103, 215, 450, 946)$$

będzie tajnym superrosącym ciągiem wag i niech  $p = 2003$  oraz  $a = 1289$ . Wówczas publicznym ciągiem wag jest

$$\mathbf{t} = (575, 436, 1586, 1030, 1921, 569, 721, 1183, 1570).$$

Jeśli teraz Alicja chce zaszyfrować tekst  $x = (1, 0, 1, 1, 0, 0, 1, 1, 1)$ , przystępuje do obliczeń:

$$y = 575 + 1586 + 1030 + 721 + 1183 + 1570 = 6665.$$

Po otrzymaniu tekstu zaszyfrowanego  $y$  za obliczenia zabiera się Bolek. Najpierw oblicza

$$\begin{aligned} z &= a^{-1} y \bmod p \\ &= 317 \cdot 6665 \bmod 2003 \\ &= 1643. \end{aligned}$$

Niech  $\mathbf{s} = (s_1, \dots, s_n)$  będzie superrosnącym ciągiem liczb całkowitych, niech  $p > \sum_{i=1}^n$  będzie liczbą pierwszą oraz  $1 \leq a \leq p-1$ . Dla  $1 \leq i \leq n$  określmy

$$t_i = as_i \bmod p$$

i niech  $\mathbf{t} = (t_1, \dots, t_n)$ . Przyjmijmy  $\mathcal{P} = \{0, 1\}^n$ ,  $\mathcal{C} = \{0, \dots, n(p-1)\}$  oraz

$$\mathcal{K} = \{(\mathbf{s}, p, a, \mathbf{t})\},$$

gdzie  $\mathbf{s}$ ,  $p$ ,  $a$ ,  $\mathbf{t}$  powstają w sposób opisany wyżej. Ciąg  $\mathbf{t}$  jest znany publicznie, natomiast  $p$ ,  $a$  i  $\mathbf{s}$  są tajne.

Dla  $K = (\mathbf{s}, p, a, \mathbf{t})$  definiujemy

$$e_K(x_1, \dots, x_n) = \sum_{i=1}^n x_i t_i.$$

Dla  $0 \leq y \leq n(p-1)$  definiujemy  $z = a^{-1}y \bmod p$  i rozwiążujemy problem podzbioru  $(s_1, \dots, s_n, z)$ , otrzymując  $d_K(y) = (x_1, \dots, x_n)$ .

### RYSUNEK 5.13. Kryptosystem plecakowy Merkle'a-Hellmana

Następnie rozwiązuje problem sumy podzbioru dla przypadku  $I = (\mathbf{s}, z)$ , używając algorytmu opisanego na rysunku 5.12. W rezultacie otrzymuje tekst jawnny  $(1, 0, 1, 1, 0, 0, 1, 1, 1)$ .  $\square$

We wczesnych latach osiemdziesiątych ubiegłego wieku system kryptograficzny Merkle'a-Hellmana został złamany przez Shamira, który potrafił wykorzystać do tego celu algorytm programowania całkowitoliczbowego Lenstry. Daje to kryptoanalitykowi Oskarowi dostęp do klucza Bolka (lub klucza równoważnego) i w konsekwencji możliwość deszyfrowania komunikatów dokładnie tak, jak to czyni Bolek.

## 5.4. System McEliece'a

System kryptograficzny McEliece'a opiera się na tej samej zasadzie projektowej co system Merkle'a-Hellmana: deszyfrowanie jest prostym szczególnym przypadkiem pewnego problemu NP-zupełnego, który z zewnątrz wygląda jak jego postać ogólna. Teraz owym problemem NP-zupełnym jest problem dekodowania ogólnego (binarnego) kodu korekcji błędów. Wiadomo jednak, że dla wie-

lu szczególnych klas kodów istnieją algorytmy działające w czasie wielomianowym. Jedną z takich klas stanowią kody Goppa, które leżą u podstaw systemu McEliece'a.

Zacznijmy od kilku ważnych definicji.

#### DEFINICJA 5.4

Niech  $k, n$  będą dodatnimi liczbami całkowitymi,  $k \leq n$ . Kodem  $[n, k]$ , **C**, nazywamy  $k$ -wymiarową podprzestrzeń przestrzeni liniowej  $(\mathbb{Z}_2)^n$  wszystkich  $n$ -wyrazowych ciągów zero-jedynkowych.

Jeśli **C** jest kodem  $[n, k]$ , to macierzą generującą dla **C** jest macierz binarna  $k \times n$ , której wiersze stanowią bazę dla **C**.

Niech  $\mathbf{x}, \mathbf{y} \in (\mathbb{Z}_2)^n$ , gdzie  $\mathbf{x} = (x_1, \dots, x_n)$  oraz  $\mathbf{y} = (y_1, \dots, y_n)$ . Określamy *odległość Hamminga* w sposób następujący:

$$d(\mathbf{x}, \mathbf{y}) = |\{i : 1 \leq i \leq n, x_i \neq y_i\}|,$$

czyli jako liczbę współrzędnych, na których różnią się wektory  $\mathbf{x}$  i  $\mathbf{y}$ .

Niech **C** będzie kodem  $[n, k]$ . Odległością kodu **C** nazywamy wielkość

$$d(\mathbf{C}) = \min\{d(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in \mathbf{C}, \mathbf{x} \neq \mathbf{y}\}.$$

Kod  $[n, k]$  z odlegością  $d$  nazywamy *kodem*  $[n, k, d]$ .

Kod korekcji błędów służy do poprawiania losowych błędów zdarzających się podczas przekazywania danych (binarnych) za pośrednictwem kanału, w którym występują zakłóczenia. W skrócie odbywa się to w sposób następujący. Niech  $G$  będzie macierzą generującą dla pewnego kodu  $[n, k, d]$ . Założymy, że chcemy przekazać binarny串  $k$ -wyrazowy  $\mathbf{x}$ . Wówczas Alicja *koduje*  $\mathbf{x}$  jako串  $n$ -wyrazowy  $\mathbf{y} = \mathbf{x}G$  i przekazuje przez kanał串  $\mathbf{y}$ .

Dalej, Bolek otrzymuje  $n$ -wyrazowy串  $\mathbf{r}$ , niekoniecznie tożsamą z  $\mathbf{y}$ . Przystępuje do *dekodowania*  $\mathbf{r}$ , stosując strategię *najbliższego sąsiada*. Przy tej strategii Bolek znajduje słowo kodowe  $\mathbf{y}'$  o najmniejszej odległości od  $\mathbf{r}$ . Następnie dekoduje  $\mathbf{r}$  jako  $\mathbf{y}'$ , by wreszcie wyznaczyć串  $k$ -wyrazowy  $\mathbf{x}'$ , taki że  $\mathbf{y}' = \mathbf{x}'G$ . Bolek liczy na to, że  $\mathbf{y}' = \mathbf{y}$ , więc również  $\mathbf{x}' = \mathbf{x}$ . (Inaczej mówiąc, ma nadzieję, że wszystkie błędy transmisji zostały poprawione).

Można dość łatwo wykazać, że jeśli podczas transmisji wystąpiło nie więcej niż  $(d - 1)/2$  błędów, to stosując strategię najbliższego sąsiada, rzeczywiście usuwa się wszystkie błędy.

Zastanówmy się przez chwilę, jak zrealizować strategię najbliższego sąsiada w praktyce. Ponieważ  $|\mathbf{C}| = 2^k$ , więc jeśli Bolek ma porównywać  $\mathbf{r}$  z każdym słowem kodowym, musi sprawdzić  $2^k$  wektorów, co w porównaniu z  $k$  jest liczbą wykładniczo wielką. Innymi słowy, ten oczywisty algorytm nie działa w czasie wielomianowym.

Odmienne podejście do tego problemu odwołuje się do pojęcia syndromu i stanowi podstawę wielu praktycznych algorytmów dekodujących. Jeśli **C** jest kodem  $[n, k, d]$  z macierzą generującą  $G$ , to *macierzą sprawdzania parzystości* dla

kodu  $\mathbf{C}$  nazywamy macierz zero-jedynkową  $(n - k) \times n$  (nazwijmy ją  $H$ ), której wiersze stanowią bazę ortogonalnego dopełnienia  $\mathbf{C}$ , oznaczanego przez  $\mathbf{C}^\perp$  i nazywanego *kodem dualnym* do  $\mathbf{C}$ . Formułując to inaczej, wiersze macierzy  $H$  są wektorami liniowo niezależnymi, a  $GH^T$  jest macierzą zerową  $k \times (n - k)$ .

Dla danego wektora  $\mathbf{r} \in (\mathbb{Z}_2)^n$ , *syndromem*  $\mathbf{r}$  nazywamy  $H\mathbf{r}^T$ . Syndrom jest więc wektorem kolumnowym o  $n - k$  elementach.

Następujące podstawowe wyniki wynikają wprost z ogólnych twierdzeń algebry liniowej.

### TWIERDZENIE 5.2

Niech  $\mathbf{C}$  będzie kodem  $[n, k]$  z macierzą generującą  $G$  oraz macierzą sprawdzania parzystości  $H$ . Wówczas  $\mathbf{x} \in (\mathbb{Z}_2)^n$  jest słowem kodowym wtedy i tylko wtedy, gdy

$$H\mathbf{x}^T = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Ponadto, jeśli  $\mathbf{x} \in \mathbf{C}$ ,  $\mathbf{e} \in (\mathbb{Z}_2)^n$  oraz  $\mathbf{r} = \mathbf{x} + \mathbf{e}$ , to  $H\mathbf{r}^T = H\mathbf{e}^T$ .

Pomyślmy o  $\mathbf{e}$  jako o wektorze błędów powstały w trakcie transmisji słowa kodowego  $\mathbf{x}$ . Wówczas  $\mathbf{r}$  reprezentuje wektor, który dociera do odbiorcy. Powyższe twierdzenie orzeka, iż syndrom zależy wyłącznie od błędów, a nie od konkretnego przesyłanego słowa kodowego.

Sugeruje to następujące podejście do problemu dekodowania, znane jako *dekodowanie syndromu*. Po pierwsze, obliczamy  $\mathbf{s} = H\mathbf{r}^T$ . Jeśli  $\mathbf{s}$  jest wektorem zerowym, dekodujemy  $\mathbf{r}$  jako  $\mathbf{r}$ . W przeciwnym przypadku generujemy kolejno wszystkie możliwe wektory błędów o wadze 1. Dla każdego takiego wektora  $\mathbf{e}$  obliczamy  $H\mathbf{e}^T$ . Jeśli dla któregokolwiek z tych wektorów  $\mathbf{e}$  mamy  $H\mathbf{e}^T = \mathbf{s}$ , to dekodujemy  $\mathbf{r}$  jako  $\mathbf{r} - \mathbf{e}$ . W przeciwnym razie generujemy dalej wszystkie wektory błędów o wadze  $2, \dots, \lfloor(d-1)/2\rfloor$ . Jeśli w jakimkolwiek momencie mamy  $H\mathbf{e}^T = \mathbf{s}$ , to dekodujemy  $\mathbf{r}$  jako  $\mathbf{r} - \mathbf{e}$  i kończymy. W przypadku gdy taka równość nie jest nigdy spełniona, stwierdzamy, że podczas transmisji wystąpiło więcej niż  $\lfloor(d-1)/2\rfloor$  błędów.

Dzięki takiemu podejściu możemy dekodować otrzymany wektor w co najwyżej

$$1 + \binom{n}{1} + \dots + \binom{n}{\lfloor(d-1)/2\rfloor}$$

krokach.

Taka metoda działa dla dowolnego kodu liniowego. W przypadku niektórych szczególnych rodzajów kodów procedurę dekodowania można przyspieszyć, jednakże decyzyjna wersja dekodowania metodą najbliższego sąsiada jest w istocie

problemem NP-zupełnym. Nie istnieje więc żaden algorytm, który rozwiązywałby w czasie wielomianowym problem dekodowania drogą poszukiwania najbliższego sąsiada (gdy liczba błędów nie jest ograniczona z góry przez  $\lfloor(d-1)/2\rfloor$ ).

Tak jak dla problemu sumy podzbioru możemy wyróżnić pewne szczególne, „łatwe” przypadki, a następnie ukryć je tak, by wyglądały na „trudny” przypadek ogólny problemu. Zbyt wiele miejsca zajęłoby tu wejście w teorię, podsumujmy więc jedynie wyniki. McEliece zaproponował „łatwy” przypadek szczególny polegający na użyciu kodów, znanych jako *kody Goppa*. Dla tych kodów istnieje rzeczywiście efektywny algorytm dekodujący, a ponadto istnieje bardzo wiele nierównoważnych kodów Goppa o tych samych parametrach i łatwo je wygenerować.

Parametry kodów Goppa są następującej postaci:  $n = 2^m$ ,  $d = 2t + 1$  oraz  $k = n - mt$ . McEliece sugerował, by w praktycznej implementacji systemów kryptograficznych z kluczem publicznym przyjąć  $m = 10$  i  $t = 50$ . Daje to kod Goppa, który jest kodem [1024, 524, 101]. Każdy tekst jawny jest binarnym ciągiem 524-wyrazowym, a każdy tekst zaszyfrowany jest ciągiem binarnym 1024-wyrazowym. Klucz publiczny jest macierzą binarną  $524 \times 1024$ .

Rysunek 5.14 zawiera opis systemu kryptograficznego McEliece'a.

Niech  $G$  będzie macierzą generującą dla kodu **C** Goppa o parametrach  $[n, k, d]$ , gdzie  $n = 2^m$ ,  $d = 2t + 1$  oraz  $k = n - mt$ . Niech  $S$  będzie macierzą  $k \times k$  odwracalną nad  $\mathbb{Z}_2$ , niech  $P$  będzie macierzą permutacji  $n \times n$  i  $G' = SGP$ . Przyjmijmy  $\mathcal{P} = (\mathbb{Z}_2)^k$ ,  $\mathcal{C} = (\mathbb{Z}_2)^n$  oraz

$$\mathcal{K} = \{(G, S, P, G')\},$$

gdzie  $G$ ,  $S$ ,  $P$ ,  $G'$  powstają w opisany wyżej sposób. Macierz  $G'$  jest znana publicznie, natomiast  $G$ ,  $S$ ,  $P$  są tajne.

Dla  $K = (G, S, P, G')$  definiujemy

$$e_K(\mathbf{x}, \mathbf{e}) = \mathbf{x}G' + \mathbf{e},$$

gdzie  $\mathbf{e} \in (\mathbb{Z}_2)^n$  jest losowym wektorem o wadze  $t$ .

Bolek deszyfruje kryptogram  $\mathbf{y} \in (\mathbb{Z}_2)^n$  za pomocą następującego ciągu operacji:

- (1) oblicza  $\mathbf{y}_1 = \mathbf{y}P^{-1}$
- (2) dekoduje  $\mathbf{y}_1$ , otrzymując  $\mathbf{y}_1 = \mathbf{x}_1 + \mathbf{e}_1$ , gdzie  $\mathbf{x}_1 \in \mathbf{C}$
- (3) oblicza  $\mathbf{x}_0 \in (\mathbb{Z}_2)^k$ , takie że  $\mathbf{x}_0G = \mathbf{x}_1$
- (4) oblicza  $\mathbf{x} = \mathbf{x}_0S^{-1}$

**RYSUNEK 5.14.** System kryptograficzny McEliece'a

Zilustrujmy procedury kodowania i dekodowania bardzo prostym przykładem.

*Przykład 5.11*

Macierz

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

jest macierzą generującą dla kodu [7, 4, 3] znanego jako *kod Hamminga*. Przypuśćmy, że Bolek wybiera macierze

$$S = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

oraz

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

Wówczas publiczną macierzą generującą staje się macierz

$$G' = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

Załóżmy teraz, że Alicja szyfruje tekst jawny  $\mathbf{x} = (1, 1, 0, 1)$ , używając jako losowego wektora błędów o wadze 1 wektora  $\mathbf{e} = (0, 0, 0, 0, 1, 0, 0)$ . Z obliczeń wynika następujący tekst zaszyfrowany:

$$\begin{aligned} \mathbf{y} &= \mathbf{x}G' + \mathbf{e} \\ &= (1, 1, 0, 1) \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} + (0, 0, 0, 0, 1, 0, 0) \\ &= (0, 1, 1, 0, 0, 1, 0) + (0, 0, 0, 0, 1, 0, 0) \\ &= (0, 1, 1, 0, 1, 1, 0). \end{aligned}$$

Gdy Bolek otrzymuje tekst zaszyfrowany  $\mathbf{y}$ , oblicza napierw

$$\begin{aligned}\mathbf{y}_1 &= \mathbf{y}P^{-1} \\ &= (0, 1, 1, 0, 1, 1, 0) \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \\ &= (1, 0, 0, 0, 1, 1, 1).\end{aligned}$$

Następnie deszyfruje  $\mathbf{y}_1$ , otrzymując  $\mathbf{x}_1 = (1, 0, 0, 0, 1, 1, 0)$  (zauważmy, że  $\mathbf{e}_1 \neq \mathbf{e}$  w wyniku mnożenia przez  $P^{-1}$ ).

Dalej Bolek tworzy wektor  $\mathbf{x}_0 = (1, 0, 0, 0)$  (wybiera pierwsze cztery współrzędne wektora  $\mathbf{x}_1$ ) i na koniec oblicza:

$$\begin{aligned}\mathbf{x} &= S^{-1}\mathbf{x}_0 \\ &= \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} (1, 0, 0, 0) \\ &= (1, 1, 0, 1).\end{aligned}$$

To właśnie tekst jawnny zaszyfrowany przez Alicję. □

## 5.5. Uwagi i bibliografia

System kryptograficzny ElGamala opisano w [EL85]. Algorytm Pohlig–Hellmana został opublikowany w [PH78]; uwagi dotyczące pojedynczych bitów problemu logarytmu dyskretnego oparliśmy na pracy Peralty [PE86]. Artykuły LaMacchii i Odlyzki [LO91] oraz McCurleya [MC90] są godnymi poleceniami źródłem dalszych informacji o samym problemie logarytmu dyskretnego.

Pisząc o ciałach skończonych, korzystaliśmy głównie z książki Lidl i Niederreitera [LN83]. Dobry podręcznik tej tematyki napisał McEliece [MC87], natomiast Menezes i inni przedstawili wyniki badań nad zastosowaniami ciał skończonych w monografii [MBGMVY93]. W 1993 roku ukazał się artykuł Gordona i McCurleya [GM93] o problemie logarytmu dyskretnego w  $GF(2^n)$ .

Pomysł użycia krzywych eliptycznych w systemach kryptograficznych z kluczem publicznym jest autorstwa Koblitza [KO87] i Millera [Mi86]. Menezes jest autorem monografii [ME93] o systemach opartych na krzywych eliptycznych;

patrz także Menezes i Vanstone [MV93] oraz Koblitz [Ko94]. Elementarne omówienie krzywych eliptycznych można znaleźć w pracy Silvermana i Tate'a [ST92]. Przeniesienie problemu logarytmu dyskretnego z krzywych eliptycznych na ciała skończone, które zaproponowali Menezes, Okamoto i Vanstone, można znaleźć w [MOV94] (patrz także [ME93]).

Kryptosystem Merkle'a–Hellmana został przedstawiony w [MH78]. Pierwszy złamał go Shamir [SH84], natomiast „iterowaną” wersję tego systemu złamał Brickell [BR85]. Innego systemu typu plecakowego, autorstwa Chora i Rivesta [CR88], nie udało się dotąd złamać. Więcej informacji można znaleźć w artykule przeglądowym Brickella i Odlyzki [BO92].

Najważniejszym punktem odniesienia w zakresie teorii kodowania jest książka MacWilliamsa i Sloane'a [MS77]. Istnieje wiele podręczników poświęconych teorii kodowania. Są to na przykład prace Hoffmanna i innych [HLLPRW91] oraz Vanstone'a i van Oorschota [VV89]. System kryptograficzny McEliece'a został po raz pierwszy opisany w artykule [MC78]. Artykuł Chabauda [CH95] dotyczy bezpieczeństwa tego systemu.

## Ćwiczenia

- 5.1. Zaprojektuj implementację algorytmu Shanksa do obliczania logarytmów dyskretnych w  $\mathbb{Z}_p$ , gdzie  $p$  jest liczbą pierwszą, a  $\alpha$  elementem pierwotnym. Użyj swojego programu do obliczenia  $\log_{106} 12375$  w  $\mathbb{Z}_{24691}$  oraz  $\log_6 248388$  w  $\mathbb{Z}_{458009}$ .
- 5.2. Zaprojektuj implementację algorytmu Pohliga–Hellmana do obliczania logarytmów dyskretnych w  $\mathbb{Z}_p$ , gdzie  $p$  jest liczbą pierwszą, a  $\alpha$  elementem pierwotnym. Użyj swojego programu do obliczenia  $\log_5 8563$  w  $\mathbb{Z}_{28703}$  oraz  $\log_{10} 12611$  w  $\mathbb{Z}_{31153}$ .
- 5.3. Oblicz  $\log_5 896$  w  $\mathbb{Z}_{1103}$  za pomocą algorytmu przedstawionego na rysunku 5.6, przyjmując  $L_2(\beta) = 1$  dla  $\beta = 25, 219$  i  $841$  oraz  $L_2(\beta) = 0$  dla  $\beta = 163, 532, 625$  i  $656$ .
- 5.4. Odczytaj tekst zaszyfrowany w systemie ElGamala, zapisany w tablicy 5.3. Parametrami systemu są:  $p = 31847$ ,  $\alpha = 5$ ,  $a = 7899$  oraz  $\beta = 18074$ . Każdy element  $\mathbb{Z}_n$  reprezentuje trzy znaki alfabetu, tak jak w ćwiczeniu 4.6.  
Tekst jawnny pochodzi z książki Michaela Onndatje „The English Patient”, Alfred A. Knopf, Inc., New York, 1992.
- 5.5. Określ, które z następujących wielomianów są nierozkładalne nad  $\mathbb{Z}_2[x]$ :  $x^5 + x^4 + 1$ ,  $x^5 + x^3 + 1$ ,  $x^5 + x^4 + x^2 + 1$ .
- 5.6. Ciało  $GF(2^5)$  można zbudować jako  $\mathbb{Z}_2[x]/(x^5 + x^2 + 1)$ . Wykonaj w tym ciele następujące obliczenia:
  - (a) oblicz  $(x^4 + x^2)(x^3 + x + 1)$ ;
  - (b) oblicz  $(x^3 + x^2)^{-1}$  za pomocą rozszerzonego algorytmu Euklidesa;
  - (c) oblicz  $x^{25}$  za pomocą algorytmu podnoszenia do kwadratu i wymnażania.

TABLICA 5.3. Kryptogram w systemie ElGamala

(3781, 14409)	(31552, 3930)	(27214, 15442)	(5809, 30274)
(5400, 31486)	(19936, 721)	(27765, 29284)	(29820, 7710)
(31590, 26470)	(3781, 14409)	(15898, 30844)	(19048, 12914)
(16160, 3129)	(301, 17252)	(24689, 7776)	(28856, 15720)
(30555, 24611)	(20501, 2922)	(13659, 5015)	(5740, 31233)
(1616, 14170)	(4294, 2307)	(2320, 29174)	(3036, 20132)
(14130, 22010)	(25910, 19663)	(19557, 10145)	(18899, 27609)
(26004, 25056)	(5400, 31486)	(9526, 3019)	(12962, 15189)
(29538, 5408)	(3149, 7400)	(9396, 3058)	(27149, 20535)
(1777, 8737)	(26117, 14251)	(7129, 18195)	(25302, 10248)
(23258, 3468)	(26052, 20545)	(21958, 5713)	(346, 31194)
(8836, 25898)	(8794, 17358)	(1777, 8737)	(25038, 12483)
(10422, 5552)	(1777, 8737)	(3780, 16360)	(11685, 133)
(25115, 10840)	(14130, 22010)	(16081, 16414)	(28580, 20845)
(23418, 22058)	(24139, 9580)	(173, 17075)	(2016, 18131)
(19886, 22344)	(21600, 25505)	(27119, 19921)	(23312, 16906)
(21563, 7891)	(28250, 21321)	(28327, 19237)	(15313, 28649)
(24271, 8480)	(26592, 25457)	(9660, 7939)	(10267, 20623)
(30499, 14423)	(5839, 24179)	(12846, 6598)	(9284, 27858)
(24875, 17641)	(1777, 8737)	(18825, 19671)	(31306, 11929)
(3576, 4630)	(26664, 27572)	(27011, 29164)	(22763, 8992)
(3149, 7400)	(8951, 29435)	(2059, 3977)	(16258, 30341)
(21541, 19004)	(5865, 29526)	(10536, 6941)	(1777, 8737)
(17561, 11884)	(2209, 6107)	(10422, 5552)	(19371, 21005)
(26521, 5803)	(14884, 14280)	(4328, 8635)	(28250, 21321)
(28327, 19237)	(15313, 28649)		

5.7. Oto przykład implementacji kryptosystemu ElGamala w  $\text{GF}(3^3)$ . Wielomian  $x^3 + 2x^2 + 1$  jest nierozkładalny nad  $\mathbb{Z}_3[x]$ , zatem  $\mathbb{Z}_3[x]/(x^3 + 2x^2 + 1)$  jest ciałem  $\text{GF}(3^3)$ . Możemy jednoznacznie przyporządkować 26 literom alfabetu 26 niezerowych elementów ciała i w ten sposób wygodnie szyfrować zwykłe teksty. Określmy przyporządkowanie tak, aby alfabetycznemu porządkowi liter odpowiadał porządek leksykograficzny w zbiorze (niezerowych) wielomianów:

$A \leftrightarrow 1$	$B \leftrightarrow 2$	$C \leftrightarrow x$
$D \leftrightarrow x + 1$	$E \leftrightarrow x + 2$	$F \leftrightarrow 2x$
$G \leftrightarrow 2x + 1$	$H \leftrightarrow 2x + 2$	$I \leftrightarrow x^2$
$J \leftrightarrow x^2 + 1$	$K \leftrightarrow x^2 + 2$	$L \leftrightarrow x^2 + x$
$M \leftrightarrow x^2 + x + 1$	$N \leftrightarrow x^2 + x + 2$	$O \leftrightarrow x^2 + 2x$
$P \leftrightarrow x^2 + 2x + 1$	$Q \leftrightarrow x^2 + 2x + 2$	$R \leftrightarrow 2x^2$
$S \leftrightarrow 2x^2 + 1$	$T \leftrightarrow 2x^2 + 2$	$U \leftrightarrow 2x^2 + x$
$V \leftrightarrow 2x^2 + x + 1$	$W \leftrightarrow 2x^2 + x + 2$	$X \leftrightarrow 2x^2 + 2x$
$Y \leftrightarrow 2x^2 + 2x + 1$	$Z \leftrightarrow 2x^2 + 2x + 2$	

Przyjmijmy, że w systemie ElGamala Bolek używa wartości  $\alpha = x$  oraz  $a = 11$ ; wówczas  $\beta = x + 2$ . Pokaż, jak Bolek będzie deszyfrował następujący ciąg tekstu zaszyfrowanego:

$(K, H)(P, X)(N, K)(H, R)(T, F)(V, Y)(E, H)(F, A)(T, W)(J, D)(U, J)$

- 5.8. Niech  $E$  będzie krzywą eliptyczną  $y^2 = x^3 + x + 28$  określona nad  $\mathbb{Z}_{71}$ .
- Wyznacz liczbę punktów krzywej  $E$ .
  - Wykaż, że  $E$  nie jest grupą cykliczną.
  - Jaki jest maksymalny rząd elementu grupy  $E$ ? Znajdź element tego rzędu.
- 5.9. Niech  $E$  będzie krzywą eliptyczną  $y^2 = x^3 + x + 13$  określona nad  $\mathbb{Z}_{31}$ . Można wykazać, że  $\#E = 34$  oraz że  $(9, 10)$  jest elementem rzędu 34 w  $E$ . Przestrzeń tekstu jawnego w systemie kryptograficznym Menezesa–Vanstone'a nad  $E$  będzie zbiór  $\mathbb{Z}_{31}^* \times \mathbb{Z}_{31}^*$ . Założmy, że tajnym wykładnikiem Bolka jest  $a = 25$ .
- Oblicz  $\beta = aa$ .
  - Odczytaj następujący ciąg tekstu zaszyfrowanego:
- $$((4, 9), 28, 7), ((19, 28), 9, 13), ((5, 22), 20, 17), ((25, 16), 12, 27).$$
- Zakładając, że każdy tekst jawnny reprezentuje dwa znaki alfabetu, przekształć tekst jawnego w słowo języka angielskiego. (Użyjemy tu przyporządkowania  $A \leftrightarrow 1, \dots, Z \leftrightarrow 26$ , gdyż 0 nie może wystąpić w parze uporządkowanej (tekstu jawnego)).
- 5.10. Założymy, że w systemie Merkle'a–Hellmana publiczny ciąg wag wygląda następująco:
- $$\mathbf{t} = (1394, 1256, 1508, 1987, 439, 650, 724, 339, 2303, 810).$$
- Przypuśćmy, że Oskarowi udało się ustalić, że  $p = 2503$ .
- Metodą prób i błędów określ wartość  $a$ , dla której ciąg  $a^{-1}\mathbf{t} \bmod p$  jest permutacją ciągu superrosnącego.
  - Pokaż, jak można odczytać tekst zaszyfrowany 5746.
- 5.11. Można wykazać, że przedstawiona niżej macierz  $H$  jest macierzą sprawdzania parzystości dla kodu  $[15, 7, 5]$  zwanego kodem BCH.
- $$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$
- Zdekoduj, jeśli to możliwe, każdy z następujących otrzymanych wektorów  $\mathbf{r}$  za pomocą metody syndromu:
- $\mathbf{r} = (1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$ ;
  - $\mathbf{r} = (1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0)$ ;
  - $\mathbf{r} = (1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0)$ .

# 6

## Schematy podpisów

### **6.1. Wprowadzenie**

W tym rozdziale zajmiemy się *schematami podpisów*, zwany takie podpisami cyfrowymi. „Tradycyjny” podpis ręczny pod dokumentem zaświadcza o odpowiedzialności podpisanej osoby za treść. Takich podpisów używamy na co dzień: przy pisaniu listu, wypłacaniu pieniędzy z banku, podpisywaniu kontraktu itd.

Schemat podpisu jest metodą podpisywania wiadomości przechowywanej w postaci elektronicznej, a więc takiej, która może być przesyłana siecią komputerową. Omówimy tu różne schematy podpisów, zaczniemy jednak od kilku uwag o różnicach między tradycyjnym podpisem ręcznym a podpisem cyfrowym.

Pierwsza sprawa to składanie podpisu. Podpis ręczny stanowi fizycznie część podpisywanego dokumentu. Podpis cyfrowy nie jest jednak fizycznie związany z wiadomością, której dotyczy, zatem odpowiedni algorytm musi go w jakiś sposób z nią połączyć.

Drugi problem to potwierdzenie autentyczności podpisu. Podpis złożony na papierze można porównać z innymi podpisami, których autentyczność jest potwierdzona. Na przykład, gdy podpisujemy potwierdzenie transakcji dokonanej kartą kredytową, sprzedawca powinien porównać podpis składany w jego obecności z tym, który jest na odwrocie karty. Oczywiście, nie jest to sposób w pełni bezpieczny, gdyż nie jest trudno sfałszować cudzy podpis. Z kolei podpisy cyfrowe można weryfikować za pomocą publicznie znanego algorytmu weryfikacji, może więc to zrobić każdy. Użycie bezpiecznego schematu podpisu zapobiegnie możliwości fałszerstwa.

Kolejną zasadniczą różnicą między podpisem tradycyjnym a podpisem cyfrowym to fakt, iż „kopią” podpisanej wiadomości elektronicznej jest identyczna z oryginałem, podczas gdy zazwyczaj kopią podpisanego dokumentu papierowego różni się od pierwowzoru. Ta cecha dokumentów elektronicznych zmusza do podejmowania działań uniemożliwiających ponowne użycie wiadomości cyfrowej. Na przykład, gdy Bolek podpisuje wiadomość elektroniczną (czek) upoważniającą Alicję do pobrania pewnej kwoty z jego konta, chce by Alicja mogła to

zrobić tylko ten jeden raz. Tak więc sama wiadomość powinna zawierać treści, na przykład datę, zapobiegającą jej ponownemu użyciu.

Schemat podpisu składa się z dwóch składników: *algorytmu podpisu* oraz *algorytmu weryfikacji*. Bolek może podpisać wiadomość  $x$ , używając (tajnego) algorytmu  $\text{sig}$ . Uzyskany w ten sposób podpis  $\text{sig}(x)$  można następnie zweryfikować za pomocą publicznego algorytmu weryfikacji  $\text{ver}$ . Dla danej pary  $(x, y)$  algorytm weryfikacji daje odpowiedź „tak” lub „nie”, potwierdzającą lub obalającą autentyczność podpisu.

Oto formalna definicja schematu podpisu.

### DEFINICJA 6.1

*Schematem podpisu* jest piątka uporządkowana  $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$  spełniająca następujące warunki:

- (1)  $\mathcal{P}$  jest skończonym zbiorem możliwych wiadomości.
- (2)  $\mathcal{A}$  jest skończonym zbiorem możliwych podpisów.
- (3)  $\mathcal{K}$ , przestrzeń kluczy, jest skończonym zbiorem możliwych kluczy.
- (4) Dla każdego  $K \in \mathcal{K}$  istnieje *algorytm podpisu*  $\text{sig}_K \in \mathcal{S}$  oraz odpowiadający mu *algorytm weryfikacji*  $\text{ver}_K \in \mathcal{V}$ . Algorytmy  $\text{sig}_K : \mathcal{P} \rightarrow \mathcal{A}$  oraz  $\text{ver}_K : \mathcal{P} \times \mathcal{A} \rightarrow \{\text{tak, nie}\}$  są funkcjami, takimi że dla każdej wiadomości  $x \in \mathcal{P}$  i każdego podpisu  $y \in \mathcal{A}$  spełnione jest równanie

$$\text{ver}(x, y) = \begin{cases} \text{tak,} & \text{gdy } y = \text{sig}(x), \\ \text{nie,} & \text{gdy } y \neq \text{sig}(x). \end{cases}$$

Funkcje  $\text{sig}_K$  i  $\text{ver}_K$  powinny być dla każdego  $K$  funkcjami działającymi w czasie wielomianowym. Druga z nich,  $\text{ver}_K$ , jest funkcją publiczną, natomiast  $\text{sig}_K$  jest tajna. Co więcej, „podrobienie” podpisu Bolka w wiadomości  $x$  powinno być dla Oskara obliczeniowo niewykonalne. Inaczej mówiąc, dla danego  $x$  tylko Bolek może obliczyć podpis  $y$ , taki że  $\text{ver}_K(x, y) = \text{tak}$ . Schemat podpisu nie może jednak być bezwarunkowo bezpieczny, ponieważ Oskar może dla danego  $x$  wypróbować wszystkie możliwe podpisy  $y$ , korzystając w tym celu z publicznego algorytmu  $\text{ver}$ , dopóki nie natrafi na właściwy. Dysponując więc dostatecznie długim czasem, Oskar może zawsze sfałszować podpis Bolka. Dlatego tu, tak jak w przypadku systemów kryptograficznych z kluczem publicznym, chodzi o to, by znaleźć schemat podpisu obliczeniowo bezpieczny.

Pierwszym przykładem schematu podpisu niech będzie system kryptograficzny RSA z kluczem publicznym, który może być użyty do generowania podpisów cyfrowych. Przedstawiamy go na rysunku 6.1.

Tak więc Bolek podpisuje wiadomość  $x$ , korzystając z reguły deszyfrowania  $d_K$  systemu RSA. Jest on jedyną osobą, która jest w stanie to uczynić, albowiem funkcja  $d_K = \text{sig}_K$  jest tajna. Algorytm weryfikacji opiera się na regule szyfrowania  $e_K$  systemu RSA. Każdy może sprawdzić podpis, funkcja  $e_K$  jest bowiem publicznie znana.

Niech  $n = pq$  dla liczb pierwszych  $p$  i  $q$ . Przyjmijmy  $\mathcal{P} = \mathcal{A} = \mathbb{Z}_n$  i określmy

$$\mathcal{K} = \{(n, p, q, a, b) : n = pq, p, q \text{ pierwsze}, ab \equiv 1 \pmod{\phi(n)}\}.$$

Wartości  $n$  i  $b$  są znane publicznie, natomiast  $p$ ,  $q$ ,  $a$  są tajne.

Dla  $K = (n, p, q, a, b)$  definiujemy

$$\text{sig}_K(x) = x^a \pmod{n}$$

oraz

$$\text{ver}_K(x, y) = \text{tak} \Leftrightarrow x \equiv y^b \pmod{n}$$

$$(x, y \in \mathbb{Z}_n).$$

#### RYSUNEK 6.1. Schemat podpisu RSA

Zauważmy, że każdy może też sfałszować podpis Bolka na „losowej” wiadomości  $x$ , obliczając  $x = e_K(y)$  dla pewnego  $y$ ; wtedy  $y = \text{sig}_K(x)$ . Jeden ze sposobów obejścia tej trudności polega na postawieniu wymogu, by wiadomości zawierały tyle nadmiarowości, by prawdopodobieństwo tego, iż sfałszowany podpis tego rodzaju odpowiada pewnemu „sensownemu” przekazowi  $x$  było znikome. Innym wyjściem jest użycie funkcji skrótu, które w połączeniu ze schematami podpisu wyeliminują ten typ fałszerstw (kryptograficznymi funkcjami skrótu zajmiemy się w rozdziale 7).

Wreszcie, przyjrzymy się chwilę połączeniu podpisu z procesem szyfrowania z kluczem publicznym. Przypuśćmy, że Alicja chce przesłać Bolekowi wiadomość nie tylko zaszyfrowaną, ale i podpisana. Dla danego tekstu jawnego oblicza podpis  $y = \text{sig}_{\text{Alicja}}(x)$ , po czym szyfruje zarówno  $x$ , jak i  $y$ , używając do tego celu publicznej funkcji szyfrowania Bolka  $e_{\text{Bolek}}$ . Uzyskuje tekst zaszyfrowany  $z = e_{\text{Bolek}}(x, y)$  i wysyła go do Bolka. Po otrzymaniu wiadomości  $z$  Bolek najpierw za pomocą swojej funkcji deszyfrowania  $d_{\text{Bolek}}$  deszyfruje go do postaci  $(x, y)$ , a następnie, używając publicznej funkcji weryfikacji Alicji, sprawdza czy  $\text{ver}_{\text{Alicja}}(x, y) = \text{tak}$ .

Co by się zmieniło, gdyby Alicja zaczęła od zaszyfrowania tekstu  $x$ , a dopiero później podpisała tekst zaszyfrowany? Wówczas obliczyłaby

$$z = e_{\text{Bolek}}(x) \quad \text{oraz} \quad y = \text{sig}_{\text{Alicja}}(z).$$

Parę  $(z, y)$  wysłałaby do Bolka. Ten odczytałby  $z$  jako  $x$  i sprawdziłby na  $x$  podpis  $y$  za pomocą funkcji  $\text{ver}_{\text{Alicja}}$ . Przy tym podejściu pojawia się jednak pewien potencjalny problem. Otóż gdyby Oskar przechwycił parę  $(z, y)$  tego typu, mógłby zastąpić podpis Alicji  $y$  własnym podpisem

$$y' = \text{sig}_{\text{Oskar}}(z).$$

(Zauważmy, że Oskar może podpisać tekst zaszyfrowany  $z = e_{\text{Bolek}}(x)$ , nawet nie znając tekstu jawnego  $x$ ). Jeśli teraz Oskar przekaże parę  $(z, y')$  Bolkowi, ten sprawdzi podpis za pomocą funkcji  $\text{ver}_{\text{Oskar}}$  i może wyciągnąć wniosek, iż to Oskar jest autorem tekstu jawnego  $x$ . Dlatego najczęściej zaleca się, by algorytm podpisu poprzedzał proces szyfrowania.

---

## 6.2. Schemat podpisu ElGamala

Przedstawimy teraz **schemat podpisu ElGamala**, opisany po raz pierwszy w 1985 roku. Narodowy Instytut Standardów i Technologii (*National Institute of Standards and Technology*, NIST) przyjął pewną modyfikację tego schematu jako standard podpisu cyfrowego. Schemat ElGamala został pomyślany specjalnie dla podpisów w przeciwieństwie do algorytmu RSA, który może być używany równie dobrze jako schemat podpisu i jako system kryptograficzny z kluczem publicznym.

Podobnie jak system kryptograficzny ElGamala z publicznym kluczem, także schemat podpisu ElGamala jest niedeterministyczny. Oznacza to, że danej wiadomości odpowiada wiele ważnych podpisów. Algorytm weryfikacji powinien umieć rozpoznawać każdy z nich jako autentyczny. Opis schematu podpisu ElGamala znajduje się na rysunku 6.2.

Jeśli podpis jest poprawnie zbudowany, weryfikacja zakończy się powodzeniem, gdyż

$$\begin{aligned}\beta^\gamma \gamma^\delta &\equiv \alpha^{a\gamma} \alpha^{k\delta} \pmod{p} \\ &\equiv \alpha^x \pmod{p}.\end{aligned}$$

Skorzystaliśmy tu z warunku

$$a\gamma + k\delta \equiv x \pmod{p-1}.$$

Do obliczenia podpisu Bolek używa zarówno tajnej wartości  $a$  (stanowiącej część klucza), jak i tajnej liczby losowej  $k$  (służącej do podpisania tylko jednej wiadomości  $x$ ). Do weryfikacji wystarczy tylko informacja publiczna.

Prześledźmy arytmetykę takich obliczeń na prostym przykładzie.

### Przykład 6.1

Niech  $p = 467$ ,  $\alpha = 2$ ,  $a = 127$ . Wtedy

$$\begin{aligned}\beta &= \alpha^a \pmod{p} \\ &= 2^{127} \pmod{467} \\ &= 132.\end{aligned}$$

Przypuśćmy, że Bolek chce podpisać wiadomość  $x = 100$  i wybiera losową wartość  $k = 213$  (zauważmy, że  $\text{NWD}(213, 466) = 1$  oraz  $213^{-1} \pmod{466} = 431$ ).

Niech  $p$  będzie liczbą pierwszą, taką że problem logarytmu dyskretnego jest obliczeniowo nieroziągalny w  $\mathbb{Z}_p$ , i niech  $\alpha \in \mathbb{Z}_p^*$  będzie elementem pierwotnym. Przyjmijmy  $\mathcal{P} = \mathbb{Z}_p^*$ ,  $\mathcal{A} = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$  i określmy

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}.$$

Wartości  $p$ ,  $\alpha$  i  $\beta$  są publiczne, natomiast liczba  $\alpha$  jest tajna.

Dla  $K = (p, \alpha, a, \beta)$  i (tajnej) losowej liczby  $k \in \mathbb{Z}_{p-1}$  definiujemy

$$sig_K(x, k) = (\gamma, \delta),$$

gdzie

$$\gamma = \alpha^k \pmod{p}$$

oraz

$$\delta = (x - a\gamma)k^{-1} \pmod{p-1}.$$

Dla  $x, \gamma \in \mathbb{Z}_p^*$  i  $\delta \in \mathbb{Z}_{p-1}$  definiujemy

$$ver_K(x, \gamma, \delta) = \text{tak} \Leftrightarrow \beta^\gamma \gamma^\delta \equiv \alpha^x \pmod{p}.$$

### RYSUNEK 6.2. Schemat podpisu ElGamala

Wówczas

$$\gamma = 2^{213} \pmod{467} = 29$$

oraz

$$\delta = (100 - 127 \cdot 29)431 \pmod{466} = 51.$$

Każdy może ten podpis zweryfikować, sprawdzając, że

$$132^{29} 29^{51} \equiv 189 \pmod{467}$$

oraz

$$2^{100} \equiv 189 \pmod{467}.$$

Podpis jest więc autentyczny. □

Zastanówmy się teraz nad bezpieczeństwem schematu podpisu ElGamala. Przypuśćmy, że nie znając wartości  $a$ , Oskar próbuje sfałszować podpis pod pewną wiadomością  $x$ . Jeśli wybierze wartość  $\gamma$  i będzie dla niej szukać od-

powiedniej wartości  $\delta$ , będzie musiał obliczyć logarytm dyskretny  $\log_\gamma \alpha^x \beta^{-\gamma}$ . Z drugiej strony, jeśli najpierw wybierze  $\delta$ , a potem poszuka dla niej wartości  $\gamma$ , czeka go „rozwiazanie” równania

$$\beta^\gamma \gamma^\delta \equiv \alpha^x \pmod{p}$$

ze względu na „niewiadomą”  $\gamma$ . Nie jest znana żadna praktyczna metoda rozwiązywania tego zadania, choć nie wydaje się ono związane z żadnym dobrze zbadanym problemem, takim jak problem logarytmu dyskretnego. Pozostaje jeszcze możliwość istnienia jakiegoś sposobu na jednoczesne obliczenie wartości  $\gamma$  i  $\delta$ , tak by para  $(\gamma, \delta)$  była podpisem. Nikt jeszcze takiego sposobu nie odkrył, ale z drugiej strony, nikt jeszcze nie udowodnił, że tego zrobić się nie da.

Jeśli Oskar wybiera  $\gamma$  i  $\delta$ , a następnie próbuje obliczyć  $x$ , trafia na przypadek problemu logarytmu dyskretnego, mianowicie na konieczność obliczenia  $\log_\alpha \beta^\gamma \gamma^\delta$ . Przy takim podejściu nie może więc podpisać „losowej” wiadomości. Istnieje jednak metoda umożliwiająca Oskarowi złożenie podpisu dzięki jednoczesnemu wyborowi wartości  $\gamma$ ,  $\delta$  i  $x$ . Niech  $i$  oraz  $j$  będą liczbami całkowitymi, takimi że  $0 \leq i \leq p - 2$ ,  $0 \leq j \leq p - 2$  oraz  $\text{NWD}(j, p - 1) = 1$ . Wykonajmy obliczenia:

$$\begin{aligned}\gamma &= \alpha^i \beta^j \pmod{p} \\ \delta &= -\gamma j^{-1} \pmod{p-1} \\ x &= -\gamma i j^{-1} \pmod{p-1},\end{aligned}$$

gdzie  $j^{-1}$  obliczamy modulo  $(p - 1)$  (tu wchodzi w grę założenie, że  $j$  i  $p - 1$  są względnie pierwsze).

Twierdzimy, że  $(\gamma, \delta)$  jest ważnym podpisem dla wiadomości  $x$ . Dowodzi tego sprawdzenie warunku weryfikującego:

$$\begin{aligned}\beta^\gamma \gamma^\delta &\equiv \beta^\gamma (\alpha^i \beta^j)^{-\gamma j^{-1}} \pmod{p} \\ &\equiv \beta^\gamma \alpha^{-i \gamma j^{-1}} \beta^{-\gamma} \pmod{p} \\ &\equiv \alpha^{-i \gamma j^{-1}} \pmod{p} \\ &= \alpha^x \pmod{p}.\end{aligned}$$

Zilustrujmy to przykładem.

### Przykład 6.2

Niech, jak w poprzednim przykładzie,  $p = 467$ ,  $\alpha = 2$  i  $\beta = 132$ . Przypuśćmy, że Oskar wybiera  $i = 99$  oraz  $j = 179$ ; wówczas  $j^{-1} \pmod{p-1} = 151$ . Teraz Oskar wykonuje następujące obliczenia:

$$\begin{aligned}\gamma &= 2^{99} 132^{179} \pmod{467} = 117 \\ \delta &= -117 \cdot 151 \pmod{466} = 41 \\ x &= 99 \cdot 41 \pmod{466} = 331.\end{aligned}$$

Para  $(117, 41)$  jest więc ważnym podpisem dla wiadomości  $331$ , co można sprawdzić przez następujące obliczenia:

$$132^{117} 117^{41} \equiv 303 \pmod{467}$$

oraz

$$2^{331} \equiv 303 \pmod{467}.$$

Podpis jest rzeczywiście autentyczny. □

Przedstawimy teraz inny rodzaj fałszerstwa, przy którym punktem wyjścia Oskara jest wiadomość podpisana wcześniej przez Bolka. Założymy, że  $(\gamma, \delta)$  jest ważnym podpisem wiadomości  $x$ ; wtedy Oskar jest w stanie podpisać następującą wiadomość. Niech  $h, i, j$  będą liczbami całkowitymi,  $0 \leq h, i, j \leq p - 2$  oraz  $\text{NWD}(h\gamma - j\delta, p - 1) = 1$ . Wykonajmy następujące obliczenia:

$$\begin{aligned}\lambda &= \gamma^h \alpha^i \beta^j \pmod{p} \\ \mu &= \delta \lambda (h\gamma - j\delta)^{-1} \pmod{p-1} \\ x' &= \lambda (hx + i\delta)(h\gamma - j\delta)^{-1} \pmod{p-1},\end{aligned}$$

gdzie  $(h\gamma - j\delta)^{-1}$  obliczamy modulo  $(p - 1)$ . Sprawdzenie warunku weryfikacyjnego

$$\beta^\lambda \lambda^\mu \equiv \alpha^{x'} \pmod{p}$$

jest już proste, choć żmudne. Para  $(\lambda, \mu)$  jest ważnym podpisem dla wiadomości  $x'$ .

Obie opisane wyżej metody umożliwiają sfałszowanie prawdziwych podpisów, lecz nie wydaje się, by przeciwnik był w stanie w ten sposób sfałszować podpis w wybranej przez siebie wiadomości bez uprzedniego rozwiązania problemu logarytmu dyskretnego. Tak więc żadna z tych metod, jak można sądzić, nie stanowi zagrożenia dla bezpieczeństwa schematu podpisu ElGamala.

Wspomnijmy na zakończenie o kilku sposobach złamania schematu ElGamala, gdy nie zachowano należytej ostrożności (będą to dalsze przykłady niepowodzenia protokołu omawianego w ćwiczeniach na końcu rozdziału 4). Przede wszystkim, nie wolno ujawniać losowej wartości  $k$ , używanej do obliczenia podpisu, gdyż znajomość tej liczby pozwala bez trudu obliczyć

$$a = (x - k\delta)\gamma^{-1} \pmod{p-1}.$$

Oczywiście, znając  $a$ , Oskar łamie system i może do woli fałszować podpisy.

Kolejny błąd polega na użyciu tej samej liczby  $k$  do podpisania dwóch różnych wiadomości. To także ułatwia Oskarowi obliczenie  $a$  i złamanie systemu, a może

to zrobić w sposób następujący. Założymy, że  $(\gamma, \delta_1)$  jest podpisem wiadomości  $x_1$ , a  $(\gamma, \delta_2)$  jest podpisem  $x_2$ . Mamy wówczas

$$\beta^\gamma \gamma^{\delta_1} \equiv \alpha^{x_1} \pmod{p}$$

oraz

$$\beta^\gamma \gamma^{\delta_2} \equiv \alpha^{x_2} \pmod{p}.$$

Stąd

$$\alpha^{x_1 - x_2} \equiv \gamma^{\delta_1 - \delta_2} \pmod{p}.$$

Podstawiając  $\gamma = \alpha^k$ , otrzymujemy równanie z niewiadomą  $k$ :

$$\alpha^{x_1 - x_2} \equiv \alpha^{k(\delta_1 - \delta_2)} \pmod{p},$$

które jest równoważne równaniu

$$x_1 - x_2 \equiv k(\delta_1 - \delta_2) \pmod{p-1}.$$

Niech teraz  $d = \text{NWD}(\delta_1 - \delta_2, p-1)$ . Ponieważ  $d|(p-1)$  oraz  $d|(\delta_1 - \delta_2)$ , wnioskujemy, że  $d|(x_1 - x_2)$ . Wprowadźmy oznaczenia:

$$\begin{aligned} x' &= \frac{x_1 - x_2}{d} \\ \delta' &= \frac{\delta_1 - \delta_2}{d} \\ p' &= \frac{p-1}{d}. \end{aligned}$$

Wówczas kongruencja przyjmuje postać

$$x' \equiv k\delta' \pmod{p'}.$$

Ponieważ  $\text{NWD}(\delta', p') = 1$ , możemy obliczyć

$$\epsilon = (\delta')^{-1} \pmod{p'},$$

a wtedy wartość  $k$  modulo  $p'$  wynika z równości

$$k = x'\epsilon \pmod{p'}.$$

Z tego otrzymujemy  $d$  możliwych wartości  $k$ :

$$k = x'\epsilon + ip' \pmod{p-1}$$

dla pewnego  $i$ , gdzie  $0 \leq i \leq d-1$ . Przez sprawdzenie warunku

$$\gamma \equiv \alpha^k \pmod{p}$$

można wybrać z tych  $d$  wartości jedyną poprawną.

### 6.3. Standard podpisu cyfrowego

**Standard podpisu cyfrowego** (zwany w skrócie **DSS** od ang. *Digital Signature Standard*) jest modyfikacją schematu podpisu ElGamala. Opublikowany w Rejestrze Federalnym 19 maja 1994 roku, został przyjęty jako standard 1 grudnia tego samego roku (choć propozycja pojawiła się już w sierpniu 1991 r.). Zaczniemy tu od uzasadnienia zmian wprowadzonych do schematu ElGamala, a następnie opiszemy, jak te zmiany zostały zrealizowane.

W wielu sytuacjach wiadomość podlega tylko jednokrotnemu zaszyfrowaniu i deszyfrowaniu. Do tego celu wystarczy dowolny system kryptograficzny, jeśli tylko jest on uznany za bezpieczny w momencie szyfrowania. Jednak podpisana wiadomość może funkcjonować jako dokument prawny, na przykład, jako kontrakt lub testament, co czyni bardzo prawdopodobnym, iż jeszcze wiele lat po jej podpisaniu zajdzie konieczność weryfikacji podpisu. W takiej sytuacji staje się tym bardziej istotne, by przedsięwziąć większe środki ostrożności co do bezpieczeństwa schematu podpisu niż w przypadku systemu kryptograficznego. Schemat podpisu ElGamala nie jest bardziej bezpieczny niż problem logarytmu dyskretnego, wymaga więc użycia dużego modułu  $p$ . Z pewnością liczba  $p$  powinna mieć co najmniej 512 bitów, choć wielu specjalistów uznałoby, że dopiero 1024 bity zapewniają bezpieczeństwo w dającej się przewidzieć przyszłości.

Jednakże nawet 512-bitowy moduł prowadzi do 1024-bitowego podpisu. W licznych potencjalnych zastosowaniach, z których wiele wiąże się z użyciem inteligentnych kart, pożądany byłby krótszy podpis. Dzięki zręcznej modyfikacji schematu ElGamala, w DSS wiadomość 160-bitowa zawiera podpis 320-bitowy, ale w obliczeniach używa się 512-bitowego modułu  $p$ . Osiąga się to przez sprowadzenie obliczeń do podgrupy grupy  $\mathbb{Z}_p^*$  zawierającej  $2^{160}$  elementów. Bezpieczeństwo schematu opiera się na założeniu, że problem znalezienia logarytmów dyskretnych w wyróżnionej podgrupie zapewnia należyté bezpieczeństwo.

Pierwsza zmiana polega na zmianie znaku „–” na „+” w definicji  $\delta$ . Tak więc

$$\delta = (x + a\gamma)k^{-1} \pmod{(p - 1)}.$$

Zmienia się wtedy także warunek weryfikacji:

$$\alpha^x \beta^\gamma \equiv \gamma^\delta \pmod{p}. \quad (6.1)$$

Jeśli  $\text{NWD}(x + a\gamma, p - 1) = 1$ , to  $\delta^{-1} \pmod{(p - 1)}$  istnieje i możemy zmodyfikować warunek (6.1) do następującej postaci:

$$\alpha^{x\delta^{-1}} \beta^{\gamma\delta^{-1}} \equiv \gamma \pmod{p}. \quad (6.2)$$

A teraz przyjrzymy się największej innowacji, jaka znalazła się w DSS. Przyjmujemy, że  $q$  jest 160-bitową liczbą pierwszą, taką że  $q|(p - 1)$ , oraz  $\alpha$  jest pierwiastkiem stopnia  $q$  z 1 modulo  $p$ . (Łatwo takie  $\alpha$  zbudować: jeśli  $\alpha_0$  jest elementem pierwotnym w  $\mathbb{Z}_p$ , to  $\alpha = \alpha_0^{(p-1)/q} \pmod{p}$ ). Wówczas  $\beta$  i  $\gamma$  także będą pierwiastkami stopnia  $q$  z 1, a w takim razie możliwa dowolne wykładniki przy  $\alpha$ ,

$\beta$  i  $\gamma$  zredukować modulo  $q$  bez zmiany warunku weryfikacji (6.2). Jest tu jednak pewien haczyk: po lewej stronie warunku (6.2)  $\gamma$  występuje w wykładniku, natomiast po prawej występuje ponownie, ale już nie w wykładniku. Tak więc jeśli redukujemy  $\gamma$  modulo  $q$ , musimy zredukować modulo  $q$  również całą lewą stronę warunku (6.2), aby przeprowadzić weryfikację. Zwróćmy uwagę, że po wykonaniu dodatkowej redukcji modulo  $q$  warunek (6.1) przestanie działać. Pełny opis DSS przedstawiamy na rysunku 6.3.

Niech  $p$  będzie 512-bitową liczbą pierwszą, taką że problem logarytmu dyskretnego jest obliczeniowo nierozerwialny w  $\mathbb{Z}_p$ , i niech  $q$  będzie 160-bitowym dzielnikiem pierwszym liczby  $p - 1$ . Niech ponadto  $\alpha \in \mathbb{Z}_p^*$  będzie pierwiastkiem stopnia  $q$  z 1 modulo  $p$ . Przyjmijmy  $\mathcal{P} = \mathbb{Z}_q^*$ ,  $\mathcal{A} = \mathbb{Z}_q \times \mathbb{Z}_q$  i określmy

$$\mathcal{K} = \{(p, q, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}.$$

Wartości  $p, q, \alpha$  i  $\beta$  są publiczne, liczba  $a$  jest tajna.

Dla  $K = (p, q, \alpha, a, \beta)$  i (tajnej) losowej liczby  $k$ ,  $1 \leq k \leq q - 1$ , definiujemy

$$sig_K(x, k) = (\gamma, \delta),$$

gdzie

$$\gamma = (\alpha^k \pmod{p}) \pmod{q}$$

oraz

$$\delta = (x + a\gamma)k^{-1} \pmod{q}.$$

Dla  $x \in \mathbb{Z}_q^*$  i  $\gamma, \delta \in \mathbb{Z}_q$  weryfikacja jest realizowana poprzez następujący ciąg operacji:

$$e_1 = x\delta^{-1} \pmod{q}$$

$$e_2 = \gamma\delta^{-1} \pmod{q}$$

$$ver_K(x, \gamma, \delta) = \text{tak} \Leftrightarrow (\alpha^{e_1}\beta^{e_2} \pmod{p}) \pmod{q} = \gamma.$$

### RYSUNEK 6.3. Standard podpisu cyfrowego

Zauważmy, że musi być spełniony warunek  $\delta \not\equiv 0 \pmod{q}$ , gdyż wartość  $\delta^{-1} \pmod{q}$  jest niezbędna do weryfikacji podpisu (widać tu analogię do warunku  $\text{NWD}(\delta, p - 1) = 1$ , potrzebnego do przekształcenia (6.1) w (6.2)). Jeśli w algorytmie podpisu Bolek uzyska wartość  $\delta \equiv 0 \pmod{q}$ , musi ją odrzucić i zbudować nowy podpis, z nową wartością losową  $k$ . Należy tu stwierdzić, że nie

powinno to powodować trudności w praktyce: prawdopodobieństwo spełnienia warunku  $\delta \equiv 0 \pmod{q}$  powinno być rzędu  $2^{-160}$ , a więc można oczekwać, że prawie nigdy do tego nie dojdzie.

Oto prosty przykład dla ilustracji.

### Przykład 6.3

Przyjmijmy  $q = 101$  i  $p = 78q + 1 = 7879$ . Liczba 3 jest elementem pierwotnym w  $\mathbb{Z}_{7879}$ , zatem możemy взять

$$\alpha = 3^{78} \pmod{7879} = 170.$$

Niech  $a = 75$ . Wtedy

$$\beta = \alpha^a \pmod{7879} = 4567.$$

Załóżmy teraz, że Bolek chce podpisać wiadomość  $x = 22$  i wybiera losową wartość  $k = 50$ , z czego otrzymuje

$$k^{-1} \pmod{101} = 99.$$

Wówczas

$$\begin{aligned}\gamma &= (170^{50} \pmod{7879}) \pmod{101} \\ &= 2518 \pmod{101} \\ &= 94\end{aligned}$$

oraz

$$\begin{aligned}\delta &= (22 + 75 \cdot 94)99 \pmod{101} \\ &= 97.\end{aligned}$$

Za pomocą kolejnych obliczeń weryfikuje podpis  $(94, 97)$  pod wiadomością 22:

$$\begin{aligned}\delta^{-1} &= 97^{-1} \pmod{101} = 25 \\ e_1 &= 22 \cdot 25 \pmod{101} = 45 \\ e_2 &= 94 \cdot 25 \pmod{101} = 27 \\ (170^{45}4567^{27} \pmod{7879}) \pmod{101} &= 2518 \pmod{101} = 94.\end{aligned}$$

Podpis jest więc ważny. □

Gdy w 1991 roku zaproponowano DSS, pojawiło się wiele głosów krytycznych. Zastrzeżenia budziły między innymi fakt, iż proces wyboru standardu przez NIST nie miał charakteru publicznego. Standard powstał w Agencji Bezpieczeństwa Narodowego (*National Security Agency*, NSA) bez udziału amerykańskiego przemysłu. Niezależnie od zalet nowego standardu, opór budził brak przejrzystości całego postępowania.

Spośród zastrzeżeń technicznych najpoważniejszy wiązał się z ustaleniem wielkości modułu  $p$  na 512 bitów. W wielu wypowiedziach przewijał się pogląd, że byłoby lepiej, gdyby wielkość mogła być zmieniona, tak aby można było w razie potrzeby używać również modułów większych. W odpowiedzi na tę krytykę NIST zmienił specyfikację standardu, tak aby dopuścić pewną rozmaitość wielkości modułu; dopuszczono mianowicie wszystkie rozmiary modułu, które są wielokrotnościami 64 w przedziale od 512 bitów do 1024 bitów.

Kolejne uwagi krytyczne w stosunku do DSS odnosiły się do tego, że proces generowania podpisu jest znacznie szybszy od procesu jego weryfikacji. W przeciwieństwie do tego schematu, gdy używa się systemu RSA jako schematu podpisu i wykładek publicznej weryfikacji jest bardzo mały (na przykład 3), to weryfikacja odbywa się w czasie znacznie krótszym niż podpisywanie. Ta różnica skłania do rozważenia kilku kwestii wiążących się z potencjalnymi zastosowaniami schematu podpisu.

- (1) Wiadomość podpisuje się tylko raz, natomiast wielokrotnie na przestrzeni lat może się okazać konieczna weryfikacja podpisu. Wskazuje to na celowość tworzenia szybszego algorytmu weryfikacji.
- (2) Jakiego typu komputery mogą być wykorzystywane w procesie podpisywania i weryfikacji? W wielu potencjalnych zastosowaniach mogą być używane inteligentne karty o ograniczonej mocy obliczeniowej, komunikujące się z potężniejszym komputerem. Można więc myśleć o zaprojektowaniu takiego schematu, który zmniejszałby zakres obliczeń wykonywanych przez kartę. Można jednak także wyobrazić sobie sytuacje, w których inteligentna karta generuje podpis, jak i takie, w których weryfikuje podpisy, trudno więc o jednoznaczne rozstrzygnięcie.

W kwestii czasu generowania podpisu i jego weryfikowania stanowisko NIST sprowadza się do tego, że w gruncie rzeczy nie jest istotne, co jest szybsze, jeśli tylko oba procesy realizowane są wystarczająco szybko.

---

## 6.4. Jednorazowe podpisy

W tym podrozdziale zajmiemy się koncepcyjnie prostą konstrukcją schematu podpisu jednorazowego (ang. *one-time signature scheme*) z użyciem dowolnej funkcji jednokierunkowej. Termin „jednorazowy” oznacza, że podpis możnałożyć tylko pod jedną wiadomością (choć, rzecz jasna, weryfikacja może być prowadzana wielokrotnie).

Opis schematu, znanego jako **schemat podpisu Lamporta**, znajduje się na rysunku 6.4.

Opiszmy nieformalnie działanie tego schematu. Podpisywana wiadomość jest binarnym ciągiem  $k$ -wyrazowym. Każdy bit podpisuje się oddzielnie: wartość

Niech  $k$  będzie dodatnią liczbą całkowitą i  $\mathcal{P} = \{0, 1\}^k$ . Założymy, że  $f : Y \rightarrow Z$  jest funkcją jednokierunkową i niech  $A = Y^k$ . Dla losowo wybranych elementów  $y_{i,j} \in Y$  niech  $z_{i,j} = f(y_{i,j})$ ,  $1 \leq i \leq k$ ,  $j = 0, 1$ . Klucz  $K$  składa się z  $2k$  elementów  $y$  i  $2k$  elementów  $z$ . Wartości  $y$  są tajne, podczas gdy wartości  $z$  są publiczne.

Dla  $K = (y_{i,j}, z_{i,j} : 1 \leq i \leq k, j = 0, 1)$  definiujemy

$$\text{sig}_K(x_1, \dots, x_k) = (y_{1,x_1}, \dots, y_{k,x_k})$$

oraz

$$\text{ver}_K(x_1, \dots, x_k, a_1, \dots, a_k) = \text{tak} \Leftrightarrow f(a_i) = z_{i,x_i}, 1 \leq i \leq k.$$

#### RYSUNEK 6.4. Schemat podpisu Lamporta

$z_{i,j}$  odpowiada  $i$ -temu bitowi wiadomości o wartości  $j$  ( $j = 0, 1$ ). Każda z liczb  $z_{i,j}$  jest obrazem  $y_{i,j}$  ze względu na pewną jednokierunkową funkcję  $f$ . Podpisem  $i$ -tego bitu wiadomości jest przeciobraz  $y_{i,j}$  wartości  $z_{i,j}$  odpowiadającej  $i$ -temu bitowi. Weryfikacja polega po prostu na sprawdzeniu, że każdy element podpisu jest przeciobrazem odpowiedniego elementu klucza publicznego.

Ilustrację schematu przeprowadzimy na jednej z możliwych implementacji, używając funkcji wykładniczej  $f(x) = \alpha^x \bmod p$ , gdzie  $\alpha$  jest elementem pierwotnym modulo  $p$ .

#### Przykład 6.4

Liczba 7879 jest pierwsza, a 3 jest elementem pierwotnym w  $\mathbb{Z}_{7879}$ . Określmy

$$f(x) = 3^x \bmod 7879.$$

Założymy, że Bolek chce podpisać wiadomość składającą się z trzech bitów i w tym celu wybiera sześć (tajnych) liczb losowych:

$$y_{1,0} = 5831$$

$$y_{1,1} = 735$$

$$y_{2,0} = 803$$

$$y_{2,1} = 2467$$

$$y_{3,0} = 4285$$

$$y_{3,1} = 6449.$$

Następnie Bolek oblicza obrazy elementów  $y$  ze względu na funkcję  $f$ :

$$z_{1,0} = 2009$$

$$z_{1,1} = 3810$$

$$z_{2,0} = 4672$$

$$z_{2,1} = 4721$$

$$z_{3,0} = 268$$

$$z_{3,1} = 5731.$$

Te wartości  $z$  są publikowane. Teraz Bolek chce podpisać wiadomość

$$x = (1, 1, 0).$$

Dla tej wiadomości podpisem jest trójka

$$(y_{1,1}, y_{2,1}, y_{3,0}) = (735, 2467, 4285).$$

Aby go zweryfikować, wystarczy wykonać następujące obliczenia:

$$3^{735} \bmod 7879 = 3810$$

$$3^{2467} \bmod 7879 = 4721$$

$$3^{4285} \bmod 7879 = 268.$$

Podpis jest więc ważny. □

Oskar nie jest w stanie sfałszować podpisu, ponieważ nie potrafi odwrócić jednokierunkowej funkcji  $f$ , by otrzymać tajne wartości  $y$ . Jednakże schematu podpisu można użyć tylko raz. Gdyby Oskar miał do dyspozycji podpisy pod dwiema różnymi wiadomościami, nie sprawiłoby mu (na ogół) trudności zbudowanie podpisów (różnych od tych dwóch) do następnych wiadomości.

Przypuśćmy na przykład, że wiadomości  $(0, 1, 1)$  oraz  $(1, 0, 1)$  podpisano za pomocą tego samego schematu. Podpisem wiadomości  $(0, 1, 1)$  będzie trójka  $(y_{1,0}, y_{2,1}, y_{3,1})$ , natomiast podpisem wiadomości  $(1, 0, 1)$  trójka  $(y_{1,1}, y_{2,0}, y_{3,1})$ . Mając te dwa podpisy, Oskar może wyprodukować podpisy do wiadomości  $(1, 1, 1)$  (a mianowicie  $(y_{1,1}, y_{2,1}, y_{3,1})$ ) oraz  $(0, 0, 1)$  (trójka  $(y_{1,0}, y_{2,0}, y_{3,1})$ ).

Mimo swej elegancji schemat nie jest użyteczny w praktyce z powodu długości podpisów. Jeśli, na przykład, używamy modularnej funkcji wykładniczej, tak jak w ostatnim przykładzie, to bezpieczna implementacja wymagałaby co najmniej 512-bitowej liczby  $p$ , co pociąga za sobą podpisywanie każdego bitu wiadomości ciągiem 512-bitowym. W rezultacie podpis byłby 512 razy dłuższy od samej wiadomości!

Zatrzymajmy się teraz przy pewnej pochodzącej od Bosa i Chauma modyfikacji, która umożliwia pewne skrócenie podpisów bez utraty bezpieczeństwa. W schemacie Lamporta powodem uniemożliwiającym Oskarowi sfałszowanie podpisu w (drugiej) wiadomości mimo znajomości podpisu w innej jest

fakt, że wartości  $y$  odpowiadające pierwszej z nich nigdy nie stanowią podzbioru zbioru wartości  $y$  odpowiadających drugiej.

Załóżmy, że mamy rodzinę  $\mathcal{B}$  podzbiorów zbioru  $B$ , taką że  $B_1 \subseteq B_2$  tylko wtedy, gdy  $B_1 = B_2$ , dla dowolnych  $B_1, B_2 \in \mathcal{B}$ . Mówimy wówczas, że  $\mathcal{B}$  ma własność Spernera. Wiadomo, że jeśli  $B$  jest zbiorem parzystej mocy  $2n$ , to maksymalna liczba elementów rodziny  $\mathcal{B}$  podzbiorów zbioru  $B$ , mającej własność Spernera, jest równa  $\binom{2n}{n}$ . Tę liczbę łatwo osiągnąć, jeśli weźmiemy wszystkie podzbiory  $n$ -elementowe zbioru  $B$ ; oczywiście żaden podzbiór  $n$ -elementowy nie jest zawarty w różnym od niego podzbiорze  $n$ -elementowym.

Przypuśćmy teraz, że tak jak poprzednio chcemy podpisać wiadomość  $k$ -bitową i wybieramy  $n$  na tyle duże, by

$$2^k \leq \binom{2n}{n}.$$

Niech  $|B| = 2n$  i niech  $\mathcal{B}$  oznacza rodzinę wszystkich  $n$ -elementowych podzbioryów zbioru  $B$ . Założymy, że  $\phi: \{0,1\}^k \rightarrow \mathcal{B}$  będzie publicznie znaną funkcją różnowartościową. Możemy wówczas każdej wiadomości przyporządkować  $n$ -elementowy podzbiór z rodziny  $\mathcal{B}$ . Będziemy mieli  $2n$  wartości  $y$  oraz  $2n$  wartości  $z$ , a podpis każdej wiadomości będzie składał się z  $n$  wartości  $y$ . Pełny opis schematu Bosa-Chauma przedstawiamy na rysunku 6.5.

Niech  $k$  będzie dodatnią liczbą całkowitą i  $\mathcal{P} = \{0,1\}^k$ . Niech dalej  $n$  będzie liczbą całkowitą, taką że  $2^k \leq \binom{2n}{n}$ ,  $B$  – zbiorem  $2n$ -elementowym. Założymy, że

$$\phi: \{0,1\}^k \rightarrow \mathcal{B}$$

jest funkcją różnowartościową, gdzie  $\mathcal{B}$  jest zbiorem wszystkich  $n$ -elementowych podzbioryów  $B$ . Niech  $f: Y \rightarrow Z$  będzie funkcją jedno-kierunkową i  $A = Y^n$ . Dla losowego elementu  $y_i \in Y$  niech  $z_i = f(y_i)$ ,  $1 \leq i \leq 2n$ . Klucz  $K$  składa się z  $2n$  elementów  $y$  i  $2n$  elementów  $z$ . Wartości  $y$  są tajne, natomiast wartości  $z$  są publiczne.

Dla  $K = (y_i, z_i, 1 \leq i \leq 2n)$  definiujemy

$$sig_K(x_1, \dots, x_k) = \{y_j : j \in \phi(x_1, \dots, x_k)\}$$

oraz

$$\begin{aligned} ver_K(x_1, \dots, x_k, a_1, \dots, a_n) &= \text{tak} \\ \Leftrightarrow \{f(a_i) : 1 \leq i \leq n\} &= \{z_j : j \in \phi(x_1, \dots, x_k)\}. \end{aligned}$$

RYSUNEK 6.5. Schemat podpisu Bosa-Chauma

Przewaga schematu Bosa–Chauma nad schematem Lamporta polega na tym, że ten pierwszy generuje krótsze podpisy. Na przykład, założymy, że chcemy podpisać wiadomość złożoną z sześciu bitów (a więc  $k = 6$ ). Wystarczy przyjąć  $n = 4$ , ponieważ  $2^6 = 64$  oraz  $\binom{8}{4} = 70$ . Dzięki temu możemy podpisać wiadomość 6-bitową za pomocą czterech wartości  $y$ , w porównaniu z sześcioma w schemacie Lamporta.

Podobnie krótszy jest klucz, składający się z ośmiu wartości  $z$ , podczas gdy w schemacie Lamporta klucz zawiera dwanaście bitów.

Schemat Bosa–Chauma wymaga funkcji różnowartościowej  $\phi$ , która przyporządkowuje każdemu binarnemu ciągowi  $k$ -wyrazowemu  $x = (x_1, \dots, x_k)$  pewien  $n$ -elementowy podzbiór zbioru  $2n$ -elementowego. Na rysunku 6.6 przedstawiamy prosty algorytm, który to przyporządkowanie realizuje.

(1)	$x = \sum_{i=1}^k x_i 2^{i-1}$
(2)	$\phi(x) = \emptyset$
(3)	$t = 2n$
(4)	$e = n$
(5)	<b>while</b> $t > 0$ <b>do</b>
(6)	$t = t - 1$
(7)	<b>if</b> $x > \binom{t}{e}$ <b>then</b>
(8)	$x = x - \binom{t}{e}$
(9)	$e = e - 1$
(10)	$\phi(x) = \phi(x) \cup \{t + 1\}$

RYSUNEK 6.6. Obliczanie  $\phi$  w schemacie Bosa–Chauma

Na przykład, zastosowanie tego algorytmu do wiadomości  $x = (0, 1, 0, 0, 1, 1)$  daje

$$\phi(x) = \{2, 4, 6, 8\}.$$

Jak duże jest w ogólnym przypadku  $n$  w stosunku do  $k$  w schemacie Bosa–Chauma? Wiemy, że musi być spełniona nierówność  $2^k \leq \binom{2n}{n}$ . Jeśli wspólny czynnik dwumianowy

$$\binom{2n}{n} = \frac{(2n)!}{(n!)^2}$$

oszacujemy zgodnie z wzorem Stirlinga, otrzymamy  $2^{2n}/\sqrt{\pi n}$ . Po pewnych uproszczeniach dochodzimy do nierówności

$$k \leq 2n - \frac{\log_2(n\pi)}{2}.$$

Asymptycznie  $n$  jest w przybliżeniu równe  $k/2$ , a więc w schemacie Bosa–Chauma uzyskujemy niemal 50% zmniejszenie długości podpisu.

## 6.5. Podpisy niezaprzeczalne

W 1989 roku Chaum i van Antwerpen wprowadzili pojęcie podpisu niezaprzecjalnego (ang. *undeniable signature*). Miało ono wiele cech nowatorskich. Najistotniejsza z nich to konieczność udziału podpisującego, Bolka, w procesie weryfikacji podpisu. Chroni go to przed powielaniem i rozsyłaniem podpisanych przez niego dokumentów bez jego zgody. Weryfikację realizuje się za pomocą *protokołu typu pytanie-odpowiedź*.

Jeśli jednak współudział Bolka jest niezbędny do potwierdzenia podpisu, to co może go powstrzymać przed wyparciem się złożonego wcześniej podpisu? Móglby przecież stwierdzić, że ważny podpis jest fałszerstwem i albo odmówić jego potwierdzenia, albo tak przeprowadzić protokół, by weryfikacja się nie powiodła. Aby zapobiec takim sytuacjom, schemat niezaprzecjalnego podpisu zawiera *protokół wyrzeczenia* (ang. *disavowal protocol*), dzięki któremu Bolek może udowodnić podrobienie podpisu. (Gdyby odmówił udziału w protokole wyrzeczenia, zostałoby to uznane za potwierdzenie autentyczności podpisu).

Tak więc schemat podpisu niezaprzecjalnego składa się z trzech składowych: algorytmu podpisu, protokołu weryfikacji oraz protokołu wyrzeczenia. Przedstawiamy najpierw algorytm podpisu oraz **protokół weryfikacji schematu podpisu niezaprzecjalnego Chauma–van Antwerpena** (rysunek 6.7).

Wyjaśnijmy znaczenie liczb  $p$  i  $q$  w tym schemacie. Schemat rozgrywa się w  $\mathbb{Z}_p$ , powinniśmy jednak również być w stanie wykonywać obliczenia w multiplikatywnej podgrupie  $G$  grupy  $\mathbb{Z}_p^*$ , której rząd jest liczbą pierwszą. W szczególności, musimy mieć możliwość obliczenia elementów odwrotnych modulo  $|G|$ , co tłumaczy, dlaczego chcemy, by  $|G|$ , rząd podgrupy, był liczbą pierwszą. Wygodnie jest przyjąć  $p = 2q + 1$  dla liczby pierwszej  $q$ . W ten sposób uzyskamy możliwie dużą podgrupę  $G$ , co jest pożądane ze względu na to, że zarówno wiadomości, jak i podpisy są elementami  $G$ .

Udowodnimy najpierw, że Alicja uzyska potwierdzenie ważnego podpisu. W poniższych obliczeniach wszystkie wykładniki redukujemy modulo  $q$ . Zauważmy na początku, że

$$d \equiv c^{a^{-1}} \pmod{p} \quad \text{czyli} \quad d \equiv y^{e_1 a^{-1}} \beta^{e_2 a^{-1}} \pmod{p}.$$

Ponieważ

$$\beta \equiv \alpha^a \pmod{p}$$

wnioskujemy, że

$$\beta^{a^{-1}} \equiv \alpha \pmod{p}.$$

Podobnie

$$y = x^a \pmod{p}$$

Niech  $p = 2q + 1$  będzie liczbą pierwszą, taką że  $q$  jest także liczbą pierwszą i problem logarytmu dyskretnego jest obliczeniowo nieroziągalny w  $\mathbb{Z}_p^*$ . Niech  $\alpha \in \mathbb{Z}_p^*$  będzie elementem rzędu  $q$  oraz  $1 \leq a \leq q-1$ . Określmy  $\beta = \alpha^a \pmod p$ . Niech dalej  $G$  oznacza multiplikatywną podgrupę rzędu  $q$  grupy  $\mathbb{Z}_p^*$  ( $G$  składa się z reszt kwadratowych modulo  $p$ ). Przyjmijmy  $\mathcal{P} = \mathcal{A} = G$  i określmy

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod p\}.$$

Wartości  $p, \alpha$  i  $\beta$  są publiczne, liczba  $a$  jest tajna.

Dla  $K = (p, \alpha, a, \beta)$  i  $x \in G$  definiujemy

$$y = \text{sig}_K(x) = x^a \pmod p.$$

Dla  $x, y \in G$  weryfikacja przebiega zgodnie z następującym protokołem:

- (1) Alicja wybiera losowo  $e_1, e_2, e_1, e_2 \in \mathbb{Z}_q^*$
- (2) Alicja oblicza  $c = y^{e_1} \beta^{e_2} \pmod p$  i przekazuje wynik Bolkowi
- (3) Bolek oblicza  $d = c^{a^{-1} \pmod q} \pmod p$  i przekazuje wynik Alicji
- (4) Alicja akceptuje ważność podpisu  $y$  wtedy i tylko wtedy, gdy

$$d \equiv x^{e_1} \alpha^{e_2} \pmod p$$

**RYSUNEK 6.7.** Schemat podpisu niezaprzeczalnego Chauma-van Antwerpena

implikuje

$$y^{a^{-1}} \equiv x \pmod p.$$

W konsekwencji mamy

$$d \equiv x^{e_1} \alpha^{e_2} \pmod p,$$

tak jak chcieliśmy.

Oto prosty przykład.

### Przykład 6.5

Przyjmijmy  $p = 467$ . Liczba 2 jest elementem pierwotnym, więc  $2^2 = 4$  jest generatorem podgrupy  $G$  reszt kwadratowych modulo 467. Możemy więc przyjąć  $\alpha = 4$ . Niech  $a = 101$ . Wówczas

$$\beta = \alpha^a \pmod{467} = 449.$$

Bolek złoży pod wiadomością  $x = 119$  podpis

$$y = 119^{101} \bmod 467 = 129.$$

Przypuśćmy teraz, że Alicja chce zweryfikować podpis  $y$  i wybiera w tym celu losowe wartości  $e_1 = 38$ ,  $e_2 = 397$ . Obliczy  $c = 13$ , na co Bolek odpowie jej wartością  $d = 9$ . Alicja weryfikuje odpowiedź, sprawdzając, że

$$119^{38}4^{397} \equiv 9 \pmod{467}.$$

Stwierdza zatem, że podpis jest ważny.  $\square$

Wykażemy następnie, że prawdopodobieństwo tego, że Bolek oszuka Alicję, kążąc jej uznać sfalszowany podpis za ważny, jest znikome. Wynik ten nie zależy od żadnych wcześniejszych założeń obliczeniowych, co oznacza, że bezpieczeństwo jest bezwarunkowe.

### TWIERDZENIE 6.1

Jeśli  $y \not\equiv x^a \pmod{p}$ , to prawdopodobieństwo uznania przez Alicję podpisu  $y$  za ważny dla wiadomości  $x$  jest równe  $1/q$ .

**DOWÓD** Zauważmy po pierwsze, że każde możliwe pytanie  $c$  odpowiada dokładnie  $q$  parom uporządkowanym  $(e_1, e_2)$  (ponieważ  $y$  i  $\beta$  są elementami grupy multiplikatywnej  $G$ , której rząd  $q$  jest liczbą pierwszą). Gdy Bolek dostaje pytanie  $c$ , nie jest w stanie ustalić, której z  $q$  możliwych par  $(e_1, e_2)$  użyła Alicja do konstrukcji liczby  $c$ . Twierdzimy, że jeśli  $y \not\equiv x^a \pmod{p}$ , to każda odpowiedź  $d \in G$  udzielona przez Bolka jest zgodna z dokładnie jedną z  $q$  par  $(e_1, e_2)$ .

Każdy element podgrupy  $G$  możemy przedstawić w postaci potęgi elementu  $\alpha$  (z wykładnikiem wyznaczonym jednoznacznie modulo  $q$ ), gdyż  $\alpha$  jest generatorem  $G$ . Stąd dla pewnych  $i, j, k, \ell \in \mathbb{Z}_q$  mamy  $c = \alpha^i$ ,  $d = \alpha^j$ ,  $x = \alpha^k$  oraz  $y = \alpha^\ell$ , przy czym stosujemy tu arytmetykę modulo  $p$ . Rozważmy następujące dwie kongruencje:

$$c \equiv y^{e_1} \beta^{e_2} \pmod{p}$$

$$d \equiv x^{e_1} \alpha^{e_2} \pmod{p}.$$

Ten układ jest równoważny następującemu:

$$i \equiv \ell e_1 + ae_2 \pmod{q}$$

$$j \equiv ke_1 + e_2 \pmod{q}.$$

Z założenia

$$y \not\equiv x^a \pmod{p}$$

wynika, że

$$\ell \not\equiv ak \pmod{q}.$$

Tak więc wyznacznik macierzy współczynników tego układu kongruencji modulo  $q$  jest różny od 0, a zatem istnieje jednoznaczne rozwiązanie układu. Inaczej mówiąc, każdy element  $d \in G$  jest poprawną odpowiedzią dla dokładnie jednej pary uporządkowanej  $(e_1, e_2)$  spośród  $q$  par możliwych. W rezultacie prawdopodobieństwo tego, że Bolek udzieli Alicji odpowiedzi  $d$ , która przejdzie test weryfikacji, jest równe  $1/q$ , co kończy dowód twierdzenia. ■

Przejdzmy teraz do protokołu wyrzeczenia. Składają się nań dwa przebiegi protokołu weryfikacji; przedstawiamy go na rysunku 6.8.

- (1) Alicja wybiera losowo  $e_1, e_2$  ( $e_1, e_2 \in \mathbb{Z}_q^*$ )
- (2) Alicja oblicza  $c = y^{e_1} \beta^{e_2} \pmod{p}$  i przekazuje wynik Bolkowi
- (3) Bolek oblicza  $d = c^{a^{-1} \pmod{q}} \pmod{p}$  i przekazuje wynik Alicji
- (4) Alicja sprawdza, że  $d \not\equiv x^{e_1} \alpha^{e_2} \pmod{p}$
- (5) Alicja wybiera losowo  $f_1, f_2$  ( $f_1, f_2 \in \mathbb{Z}_q^*$ )
- (6) Alicja oblicza  $C = y^{f_1} \beta^{f_2} \pmod{p}$  i przekazuje wynik Bolkowi
- (7) Bolek oblicza  $D = C^{a^{-1} \pmod{q}} \pmod{p}$  i przekazuje wynik Alicji
- (8) Alicja sprawdza, że  $D \not\equiv x^{f_1} \alpha^{f_2} \pmod{p}$
- (9) Alicja uzna je, że  $y$  jest sfałszowane wtedy i tylko wtedy, gdy  

$$(d\alpha^{-e_2})^{f_1} \equiv (D\alpha^{-F_2})^{e_1} \pmod{p}$$

**RYSUNEK 6.8. Protokół wyrzeczenia**

Kroki 1–4 oraz 5–8 obejmują dwa nieudane przebiegi protokołu weryfikacji. Krok 9 to „sprawdzenie zgodności”, umożliwiające Alicji stwierdzenie, czy Bolek buduje swoje odpowiedzi zgodnie z protokołem.

Niech przykład posłuży za ilustrację działania protokołu wyrzeczenia.

### Przykład 6.6

Jak poprzednio, przyjmijmy  $p = 467$ ,  $\alpha = 4$ ,  $a = 101$  oraz  $\beta = 449$ . Przypuśćmy, że wiadomość  $x = 286$  jest opatrzona podpisem  $y = 83$ , a Bolek próbuje przekonać Alicję, że podpis jest nieważny.

Załóżmy, że Alicja wybiera na początku losowe wartości  $e_1 = 45$ ,  $e_2 = 237$ . Oblicza  $c = 305$ , na co Bolek odpowiada jej liczbą  $d = 109$ . Wówczas Alicja oblicza

$$286^{45} 4^{237} \pmod{467} = 149.$$

Mamy nierówność  $149 \neq 109$ , zatem Alicja przechodzi do kroku 5 w protokole wyrzeczenia.

Niech teraz losowymi wartościami wybranymi przez Alicję będą liczby  $f_1 = 125$  i  $f_2 = 9$ . Alicja oblicza  $C = 270$ , Bolek odpowiada liczbą  $D = 68$ . Alicja liczy dalej:

$$286^{125} 4^9 \pmod{467} = 25.$$

Ponieważ  $25 \neq 68$ , Alicja przechodzi do kroku 9 protokołu i wykonuje sprawdzenie zgodności. Sprawdzenie kończy się powodzeniem, gdyż

$$(109 \cdot 4^{-237})^{125} \equiv 188 \pmod{467}$$

oraz

$$(68 \cdot 4^{-9})^{45} \equiv 188 \pmod{467}.$$

Alicja pozostaje więc w przekonaniu o nieważności podpisu.  $\square$

W tym miejscu musimy udowodnić dwie rzeczy:

- (1) Bolek potrafi przekonać Alicję, że nieważny podpis jest fałszerstwem.
- (2) Prawdopodobieństwo tego, że Bolek przekona Alicję do uwierzenia w fałszerstwo w przypadku ważnego podpisu, jest bardzo małe.

### **TWIERDZENIE 6.2**

Jeśli  $y \not\equiv x^a \pmod{p}$  oraz Alicja i Bolek postępują zgodnie z protokołem wyrzeczenia, to

$$(d\alpha^{-e_2})^{f_1} \equiv (D\alpha^{-f_2})^{e_1} \pmod{p}.$$

**DOWÓD** Z kongruencji

$$\begin{aligned} d &\equiv c^{a^{-1}} \pmod{p}, \\ c &\equiv y^{e_1} \beta^{e_2} \pmod{p} \end{aligned}$$

oraz

$$\beta \equiv \alpha^a \pmod{p}$$

wnioskujemy, że

$$\begin{aligned} (d\alpha^{-e_2})^{f_1} &\equiv \left( (y^{e_1} \beta^{e_2})^{a^{-1}} \alpha^{-e_2} \right)^{f_1} \pmod{p} \\ &\equiv y^{e_1 a^{-1} f_1} \beta^{e_2 a^{-1} f_1} \alpha^{-e_2 f_1} \pmod{p} \\ &\equiv y^{e_1 a^{-1} f_1} \alpha^{e_2 f_1} \alpha^{-e_2 f_1} \pmod{p} \\ &\equiv y^{e_1 a^{-1} f_1} \pmod{p}. \end{aligned}$$

Podobne obliczenia, w połączeniu z kongruencjami  $D \equiv C^{a^{-1}} \pmod{p}$ ,  $C \equiv y^{f_1} \beta^{f_2} \pmod{p}$  oraz  $\beta \equiv \alpha^a \pmod{p}$ , pozwalają stwierdzić, że

$$(D\alpha^{-f_2})^{e_1} \equiv y^{e_1 a^{-1} f_1} \pmod{p},$$

a więc sprawdzenie zgodności w kroku 9 daje wynik pozytywny. ■

Rozważmy teraz możliwość wyparcia się przez Bolka ważnego podpisu. W taka sytuacji odrzucamy założenie, że Bolek postępuje zgodnie z protokołem. Inaczej mówiąc, Bolek może nie wygenerować liczb  $d$  i  $D$ , tak jak wymaga tego protokołu. Dlatego w kolejnym twierdzeniu zakładamy tylko, że Bolek dostarcza wartości  $d$  i  $D$ , które spełniają warunki z kroków 4, 8 i 9 protokołu opisanego na rysunku 6.8.

### TWIERDZENIE 6.3

Załóżmy, że  $y \equiv x^a \pmod{p}$  i Alicja postępuje zgodnie z protokołem wyrzeczenia. Jeśli

$$d \not\equiv x^{e_1} \alpha^{e_2} \pmod{p}$$

oraz

$$D \not\equiv x^{f_1} \alpha^{f_2} \pmod{p},$$

to prawdopodobieństwo zdarzenia

$$(d\alpha^{-e_2})^{f_1} \not\equiv (D\alpha^{-f_2})^{e_1} \pmod{p}$$

jest równe  $1 - 1/q$ .

**DOWÓD** Założymy, że spełnione są następujące kongruencje:

$$y \equiv x^a \pmod{p}$$

$$d \not\equiv x^{e_1} \alpha^{e_2} \pmod{p}$$

$$D \not\equiv x^{f_1} \alpha^{f_2} \pmod{p}$$

$$(d\alpha^{-e_2})^{f_1} \equiv (D\alpha^{-f_2})^{e_1} \pmod{p}.$$

Pokażemy, że te warunki prowadzą do sprzeczności.

Sprawdzian zgodności można przedstawić w następującej, równoważnej postaci:

$$D \equiv d_0^{f_1} \alpha^{f_2} \pmod{p},$$

gdzie

$$d_0 = d^{1/e_1} \alpha^{-e_2/e_1} \pmod{p}$$

jest wartością zależną jedynie od kroków 1–4 protokołu.

Na mocy twierdzenia 6.1 wnosimy, że z prawdopodobieństwem  $1 - 1/q$  liczba  $y$  jest ważnym podpisem dla  $d_0$ . Zakładamy jednak, że  $y$  jest ważnym podpisem dla  $x$ , zatem z dużym prawdopodobieństwem mamy

$$x^a \equiv d_0^a \pmod{p},$$

co implikuje  $x = d_0$ .

Jednakże z założenia

$$d \not\equiv x^{e_1} \alpha^{e_2} \pmod{p}$$

wynika, że

$$x \not\equiv d^{1/e_1} \alpha^{-e_2/e_1} \pmod{p},$$

co w połączeniu z

$$d_0 \equiv d^{1/e_1} \alpha^{-e_2/e_1} \pmod{p}$$

prowadzi do wniosku, że  $x \neq d_0$ . Mamy więc sprzeczność.

Bolek może więc w ten sposób oszukać Alicję z prawdopodobieństwem  $1/q$ . ■

## 6.6. Podpisy niepodrabialne

Schemat niepodrabialnego podpisu zapewnia wzmocnione bezpieczeństwo w przypadku zaistnienia możliwości sfałszowania podpisu przez bardzo potężnego przeciwnika. W przypadku gdyby Oskar potrafił podrobić podpis Bolka, ten (z bardzo dużym prawdopodobieństwem) umiałby udowodnić fałszywość podpisu.

Opiszymy tu schemat niepodrabialnego podpisu (ang. *fail-stop signature*) zaprojektowany przez van Heysta i Pedersena w 1992 roku. Jest to schemat jednorazowy (danym kluczem można podpisać tylko jedną wiadomość). Składa się z algorytmów podpisu i weryfikacji, a także z algorytmu „dowodu fałszerstwa”. Na rysunku 6.9 przedstawiamy opis algorytmów podpisu i weryfikacji w **schemacie podpisu niepodrabialnego van Heysta–Pedersena**.

Widac od razu, że stworzony przez Bolka podpis spełni warunek weryfikacji. Przejdzmy zatem bezpośrednio do kwestii bezpieczeństwa schematu oraz do wyjaśnienia istoty niepodrabialności podpisu. Odnotujmy najpierw pewne istotne fakty dotyczące kluczów schematu. Zaczniemy od definicji. Mówimy, że dwa klucze  $(\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$  i  $(\gamma'_1, \gamma'_2, a'_1, a'_2, b'_1, b'_2)$  są *równoważne*, gdy  $\gamma_1 = \gamma'_1$  oraz  $\gamma_2 = \gamma'_2$ . Łatwo obliczyć, że w jednej klasie równoważności jest dokładnie  $q^2$  różnych kluczy.

Wykażemy na początek kilka lematów.

Niech  $p = 2q + 1$  będzie liczbą pierwszą, taką że  $q$  jest także liczbą pierwszą i problem logarytmu dyskretnego jest obliczeniowo niesrozwiązalny w  $\mathbb{Z}_p$ . Niech  $\alpha \in \mathbb{Z}_p^*$  będzie elementem rzędu  $q$  oraz  $1 \leq a_0 \leq q - 1$ . Określmy  $\beta = \alpha^{a_0} \pmod p$ . Wartości  $p, q, \alpha, \beta$  oraz  $a_0$  wybiera centralna (zaufana) władza. Wartości  $p, q, \alpha$  i  $\beta$  są publiczne i uznajemy je za niezmienne. Wartość  $a_0$  pozostaje tajna dla wszystkich (łącznie z Bolkiem).

Przyjmijmy  $\mathcal{P} = \mathbb{Z}_q$  i  $\mathcal{A} = \mathbb{Z}_q \times \mathbb{Z}_q$ . Klucz ma postać

$$K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2),$$

gdzie  $a_1, a_2, b_1, b_2 \in \mathbb{Z}_q$ ,

$$\gamma_1 = \alpha^{a_1} \beta^{a_2} \pmod p$$

oraz

$$\gamma_2 = \alpha^{b_1} \beta^{b_2} \pmod p.$$

Dla  $K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$  i  $x \in \mathbb{Z}_q$  definiujemy

$$sig_K(x) = (y_1, y_2),$$

gdzie

$$y_1 = a_1 + xb_1 \pmod q$$

oraz

$$y_2 = a_2 + xb_2 \pmod q.$$

Dla  $y = (y_1, y_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$  mamy

$$ver_K(x, y) = \text{tak} \Leftrightarrow \gamma_1 \gamma_2^x \equiv \alpha^{y_1} \beta^{y_2} \pmod p.$$

**RYSUNEK 6.9.** Schemat niepodrabialnego podpisu van Heystaa-Pedersena

#### LEMAT 6.4

Niech  $K$  i  $K'$  będą równoważnymi kluczami i założymy, że  $ver_K(x, y) = \text{tak}$ . Wtedy  $ver_{K'}(x, y) = \text{tak}$ .

**DOWÓD** Niech  $K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$  i  $K' = (\gamma_1, \gamma_2, a'_1, a'_2, b'_1, b'_2)$ , gdzie

$$\gamma_1 = \alpha^{a_1} \beta^{a_2} \pmod p = \alpha^{a'_1} \beta^{a'_2} \pmod p$$

oraz

$$\gamma_2 = \alpha^{b_1} \beta^{b_2} \pmod{p} = \alpha^{b'_1} \beta^{b'_2} \pmod{p}.$$

Załóżmy, że do podpisu  $x$  użyto klucza  $K$ , otrzymując podpis  $y = (y_1, y_2)$ , gdzie

$$y_1 = a_1 + xb_1 \pmod{q},$$

$$y_2 = a_2 + xb_2 \pmod{q}.$$

Przypuśćmy, że do sprawdzenia podpisu  $y$  stosujemy klucz  $K'$ :

$$\begin{aligned} \alpha^{y_1} \beta^{y_2} &\equiv \alpha^{a'_1+xb'_1} \beta^{a'_2+xb'_2} \pmod{p} \\ &\equiv \alpha^{a'_1} \beta^{a'_2} (\alpha^{b'_1} \beta^{b'_2})^x \pmod{p} \\ &\equiv \gamma_1 \gamma_2 x \pmod{p}. \end{aligned}$$

Tak więc użycie klucza  $K'$  również prowadzi do weryfikacji podpisu  $y$ . ■

### LEMAT 6.5

Niech  $K$  będzie kluczem i  $y = \text{sig}_K(x)$ . Wówczas istnieje dokładnie  $q$  kluczy  $K'$  równoważnych kluczowi  $K$ , takich że  $y = \text{sig}_{K'}(x)$ .

**DOWÓD** Niech  $\gamma_1$  i  $\gamma_2$  będą publicznymi składnikami klucza  $K$ . Chcemy ustalić liczbę czwórek  $(a_1, a_2, b_1, b_2)$  spełniających następujące kongruencje:

$$\gamma_1 \equiv \alpha^{a_1} \beta^{a_2} \pmod{p}$$

$$\gamma_2 \equiv \alpha^{b_1} \beta^{b_2} \pmod{p}$$

$$y_1 \equiv a_1 + xb_1 \pmod{p}$$

$$y_2 \equiv a_2 + xb_2 \pmod{p}.$$

Element  $\alpha$  jest generatorem  $G$ , zatem istnieją jednoznacznie wyznaczone wykładniki  $c_1, c_2, a_0 \in \mathbb{Z}_q$ , takie że

$$\gamma_1 \equiv \alpha^{c_1} \pmod{p}$$

$$\gamma_2 \equiv \alpha^{c_2} \pmod{p}$$

oraz

$$\beta \equiv \alpha^{a_0} \pmod{p}.$$

Tak więc warunkiem koniecznym i dostatecznym jest spełnienie poniższego układu kongruencji:

$$c_1 \equiv a_1 + a_0 a_2 \pmod{q}$$

$$c_2 \equiv b_1 + a_0 b_2 \pmod{q}$$

$$y_1 \equiv a_1 + xb_1 \pmod{q}$$

$$y_2 \equiv a_2 + xb_2 \pmod{q}.$$

Ten układ możemy z kolei przedstawić w postaci macierzowej nad  $\mathbb{Z}_q$ :

$$\begin{pmatrix} 1 & a_0 & 0 & 0 \\ 0 & 0 & 1 & a_0 \\ 1 & 0 & x & 0 \\ 0 & 1 & 0 & x \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ y_1 \\ y_2 \end{pmatrix}.$$

Jak widać, macierz współczynników układu ma rzad<sup>1</sup> 3. Istotnie, rzad jest nie mniejszy od 3, ponieważ wiersze 1, 2 i 3 są liniowo niezależne, a jednocześnie jest nie większy od 3, gdyż

$$r_1 + xr_2 - r_3 - a_0 r_4 = (0, 0, 0, 0),$$

gdzie  $r_i$  oznacza  $i$ -ty wiersz macierzy.

Układ równań ma co najmniej jedno rozwiązanie wynikające z zastosowania klucza  $K$ . Ponieważ rzad macierzy jest równy 3, zatem wymiar przestrzeni rozwiązań wynosi  $4 - 3 = 1$ , istnieje więc dokładnie  $q$  rozwiązań. Stąd wynika teza lematu. ■

Podobne rozumowanie może posłużyć do dowodu następnego wyniku. Dowód pomijamy.

### LEMAT 6.6

Niech  $K$  będzie kluczem,  $y = \text{sig}_K(x)$  oraz  $\text{ver}_K(x', y') = \text{tak}$ , gdzie  $x' \neq x$ . Wówczas istnieje co najwyżej jeden klucz  $K'$  równoważny kluczowi  $K$ , taki że  $y = \text{sig}_{K'}(x)$  i  $y' = \text{sig}_{K'}(x')$ .

Zinterpretujmy znaczenie ostatnich dwóch lematów dla bezpieczeństwa schematu. Otóż jeśli  $y$  jest ważnym podpisem wiadomości  $x$ , to istnieje  $q$  możliwych kluczy, które również spowodują przypisanie wiadomości  $x$  podpisu  $y$ . Ale jeśli  $x' \neq x$ , to z tych  $q$  kluczy otrzymamy  $q$  różnych podpisów dla wiadomości  $x'$ . Wynika stąd następujące twierdzenie.

### TWIERDZENIE 6.7

Jeśli  $\text{sig}_K(x) = y$  oraz  $x' \neq x$ , to Oskar może obliczyć  $\text{sig}_K(x')$  z prawdopodobieństwem  $1/q$ .

Zauważmy, że teza twierdzenia nie zależy od mocy obliczeniowej, jaką dysponuje Oskar. Określony w twierdzeniu poziom bezpieczeństwa wynika z tego, że Oskar nie jest w stanie stwierdzić, którego z  $q$  kluczy użył Bolek. Tak więc bezpieczeństwo ma charakter bezwarunkowy.

Zajmijmy się teraz kwestią niepodrabialności. Stwierdziliśmy dotąd, że dysponując podpisem  $y$  pod wiadomością  $x$ , Oskar nie potrafi obliczyć podpisu  $y'$  złożonego przez Bolka pod inną wiadomością  $x'$ . Można sobie wszakże wyobrazić, że Oskar umiałby obliczyć fałszywy podpis  $y'' \neq \text{sig}_K(x')$ , który pomyślnie

---

<sup>1</sup>Rzędem macierzy nazywa się maksymalną liczbę liniowo niezależnych wierszy tej macierzy.

przeszedłby weryfikację. Jednakże, jeśli Bolkowi zostanie przedłożony ważny, ale sfałszowany podpis, może on (z prawdopodobieństwem  $1/q$ ) wygenerować „dowód fałszerstwa”. Dowodem fałszerstwa jest wartość  $a_0 = \log_\alpha \beta$  znana tylko centralnej władzce.

Zakładamy więc, że Bolek dysponuje parą  $(x', y'')$ , taką że  $\text{ver}_K(x', y'') = \text{tak}$  oraz  $y'' \neq \text{sig}_K(x')$ , czyli

$$\gamma_1 \gamma_2^{x'} \equiv \alpha^{y_1''} \beta^{y_2''} \pmod{p},$$

gdzie  $y'' = (y_1'', y_2'')$ . Bolek może obliczyć własny podpis pod wiadomością  $x'$ , a mianowicie  $y' = (y_1', y_2')$ . Wtedy

$$\gamma_1 \gamma_2^{x'} \equiv \alpha^{y_1'} \beta^{y_2'} \pmod{p}.$$

Stąd

$$\alpha^{y_1''} \beta^{y_2''} \equiv \alpha^{y_1'} \beta^{y_2'} \pmod{p}.$$

Pisząc  $\beta = \alpha^{a_0} \pmod{p}$ , mamy

$$\alpha^{y_1'' + a_0 y_2''} \equiv \alpha^{y_1' + a_0 y_2'} \pmod{p},$$

lub

$$y_1'' + a_0 y_2'' \equiv y_1' + a_0 y_2' \pmod{q}.$$

Po uproszczeniu dochodzimy do warunku

$$y_1'' - y_1' \equiv a_0(y_2' - y_2'') \pmod{q}.$$

Wiemy, że  $y_2' \neq y_2'' \pmod{q}$ , gdyż  $y'$  jest fałszerstwem. Stąd  $(y_2' - y_2'')^{-1} \pmod{q}$  istnieje oraz

$$a_0 = \log_\alpha \beta = (y_1'' - y_1')(y_2' - y_2'')^{-1} \pmod{q}.$$

Oczywiście, dopuszczając taki dowód fałszerstwa, zakładamy, że Bolek sam nie potrafi obliczyć logarytmu dyskretnego  $\log_\alpha \beta$ . Jest to założenie o charakterze obliczeniowym.

Zauważmy na koniec, że w przypadku gdy dwie wiadomości zostaną podpisane z użyciem tego samego klucza  $K$ , ten klucz będzie można łatwo obliczyć. To powoduje, że mamy do czynienia ze schematem jednorazowym.

Zamykamy rozdział przykładem ilustrującym sposób dowodzenia fałszerstwa.

### Przykład 6.7

Niech  $p = 3467 = 2 \cdot 1733 + 1$ . Element  $\alpha = 4$  ma w grupie  $\mathbb{Z}_{3467}^*$  rząd 1733. Założymy, że  $a_0 = 1567$ ; wtedy

$$\beta = 4^{1567} \pmod{3467} = 514.$$

(Przypomnijmy, że Bolek zna wartości  $\alpha$  i  $\beta$ , ale nie zna  $a_0$ ). Przyjmijmy, że do konstrukcji klucza Bolek używa liczb  $a_1 = 888$ ,  $a_2 = 1024$ ,  $b_1 = 786$  oraz  $b_2 = 999$ , co daje

$$\gamma_1 = 4^{888} 514^{1024} \pmod{3467} = 3405$$

oraz

$$\gamma_2 = 4^{786} 514^{999} \pmod{3467} = 2281.$$

Przypuśćmy dalej, że Bolek dostaje wiadomość 3383 ze sfałszowanym podpisem (822, 55). Podpis jest ważny, gdyż spełniony jest warunek weryfikacji:

$$3405 \cdot 2281^{3383} \equiv 2282 \pmod{3467}$$

i

$$4^{822} 514^{55} \equiv 2282 \pmod{3467}.$$

Z drugiej strony to nie jest podpis, który mógłby wygenerować Bolek. Może on obliczyć własny podpis:

$$(888 + 3383 \cdot 786 \pmod{1733}, 1024 + 3383 \cdot 999 \pmod{1733}) = (1504, 1291).$$

Teraz przystępuje do obliczenia tajnego logarytmu dyskretnego:

$$a_0 = (822 - 1504)(1291 - 55)^{-1} \pmod{1733} = 1567.$$

To stanowi dowód fałszerstwa. □

## 6.7. Uwagi i bibliografia

Polecamy pracę Mitchella, Pipera i Wilda [MPW92] jako drobiazgowy przegląd schematów podpisu. Praca zawiera również opis dwóch metod fałszerstwowania podpisu w schemacie ElGamala, które omówiliśmy w podrozdziale 6.2.

Schemat podpisu ElGamala został opisany w pracy ElGamala [El85]. Standard podpisu cyfrowego (DSS) został po raz pierwszy opublikowany przez NIST w sierpniu 1991 roku, natomiast zyskał oficjalny status standardu w grudniu 1994 roku [NBS94]. W lipcowym numerze *Communications of the ACM* z 1992 roku znajduje się obszerne omówienie DSS i otaczających go kontrowersji. Odpowiedzi NIST na niektóre podniesione tam kwestie zamieszczono w [SB93].

Opis schematu Lamporta znajduje się w pracy Diffiego i Hellmana z 1976 roku [DH76]; o modyfikacji Bosa i Chauma można przeczytać w [BC93]. Opisany w podrozdziale 6.5 schemat podpisu niepodrabialnego pochodzi z pracy

Chauma i van Antwerpena [CvA90]. Schemat podpisu niepodrabialnego z podrozdziału 6.6 został zaczerpnięty z pracy van Heysta i Pedersena [vHP93].

Wśród przykładów dobrze znanych „złamanych” schematów podpisu należy wymienić schemat Ong–Schnorra–Shamira [OSS85] (złamany przez Estesa i innych [EAKMM86]) oraz dwuwymierny schemat permutacyjny Shamira [SH94] (złamany przez Coppersmitha, Sterna i Vaudenaya [CSV94]). Wreszcie, ESIGN jest schematem podpisu, którego autorami są Fujioka, Okamoto i Miyaguchi [FOM91]. Niektóre wersje tego schematu złamano, ale jak dotąd wariant opublikowany w [FOM91] oparł się atakom.

## Ćwiczenia

- 6.1. Przyjmijmy, że Bolek, używając schematu ElGamala, podpisuje dwie wiadomości  $x_1$  i  $x_2$  podpisami  $(\gamma, \delta_1)$  i  $(\gamma, \delta_2)$ , odpowiednio. (W obu podpisach powtarza się wartość  $\gamma$ ). Założymy także, że  $\text{NWD}(\delta_1 - \delta_2, p - 1) = 1$ .
  - (a) Opisz, jak można efektywnie obliczyć  $k$  na podstawie przedstawionej wyżej informacji.
  - (b) Opisz, jak można w tej sytuacji złamać schemat podpisu.
  - (c) Niech  $p = 31847$ ,  $\alpha = 5$  i  $\beta = 25703$ . Oblicz  $k$  i  $a$ , wiedząc, że podpis  $(23972, 31396)$  złożono pod wiadomością  $x = 8990$ , a podpis  $(23972, 20481)$  – pod wiadomością  $x = 31415$ .
- 6.2. Przyjmijmy, że implementujemy schemat podpisu ElGamala z wartościami  $p = 31847$ ,  $\alpha = 5$  oraz  $\beta = 26379$ . Napisz program komputerowy, który realizowałby następujące zadania:
  - (a) Weryfikacja podpisu  $(20679, 11082)$  pod wiadomością  $x = 20543$ .
  - (b) Ustalenie tajnego wykładnika  $a$  za pomocą metody Shanksa uzyskiwania kompromisu między czasem i pamięcią, a następnie wyznaczenie losowej wartości  $k$  użytej przy podpisywaniu wiadomości  $x$ .
- 6.3. Założymy, że Bolek stosuje schemat podpisu ElGamala tak jak w przykładzie 6.1:  $p = 467$ ,  $\alpha = 2$  oraz  $\beta = 132$ . Przypuśćmy, że do wiadomości  $x = 100$  dołączył podpis  $(29, 51)$ . Oblicz fałszywy podpis, który może wygenerować Oskar, przyjmując  $h = 102$ ,  $i = 45$  oraz  $j = 293$ . Sprawdź, że tak otrzymany podpis spełnia warunek weryfikacji.
- 6.4. Wykaż, że druga metoda fałszowania podpisu, przedstawiona w podrozdziale 6.2, również daje w wyniku podpis spełniający warunek weryfikacji.
- 6.5. Rozważmy pewien wariant schematu podpisu ElGamala. Klucz tworzymy podobnie jak poprzednio: Bolek wybiera  $\alpha \in \mathbb{Z}_p^*$  jako element pierwotny,  $a$  jest tajnym wykładnikiem ( $0 \leq a \leq p - 2$ ), takim że  $\text{NWD}(a, p - 1) = 1$ , oraz  $\beta = \alpha^a \pmod{p}$ . Kluczem jest  $K = (\alpha, a, \beta)$ , gdzie  $\alpha$  i  $\beta$  są publiczne, natomiast  $a$  jest tajne. Niech  $x \in \mathbb{Z}_p$  będzie wiadomością wymagającą podpisu. Bolek oblicza podpis  $\text{sig}_K(x) = (\gamma, \delta)$ , gdzie

$$\gamma = \alpha^k \pmod{p}$$

oraz

$$\delta = (x - k\gamma)a^{-1} \bmod (p - 1).$$

Jedyną różnicą w stosunku do oryginalnego schematu ElGamala jest sposób obliczenia  $\delta$ . Odpowiedz na następujące pytania dotyczące zmodyfikowanego schematu:

- (a) Opisz, jak można zweryfikować podpis  $(\gamma, \delta)$  pod wiadomością  $x$ , używając publicznego klucza Bolka.
  - (b) Omów przewagę obliczeniową schematu zmodyfikowanego nad oryginalnym.
  - (c) Porównaj krótko bezpieczeństwo obu schematów.
- 6.6. Przypuśćmy, że Bolek stosuje schemat DSS z wartościami  $q = 101$ ,  $p = 7879$ ,  $\alpha = 170$ ,  $a = 75$  oraz  $\beta = 4567$ , tak jak w przykładzie 6.3. Wyznacz podpis Bolka pod wiadomością  $x = 52$ , używając losowej wartości  $k = 49$ , i pokaż jak przebiega weryfikacja otrzymanego podpisu.
- 6.7. Założmy, że Bolek podpisuje dwa ciągi  $k$ -wyrazowe  $x$  i  $x'$  w schemacie Lamporta. Niech  $\ell = d(x, x')$  oznacza liczbę współrzędnych, na których oba ciągi  $x$  i  $x'$  się różnią. Wykaż, że Oskar może teraz podpisać  $2^\ell - 2$  nowych wiadomości.
- 6.8. Założmy, że w schemacie Bosa–Chauma podpisano dwie wiadomości  $x = (0, 1, 0, 0, 1, 1)$  i  $x' = (1, 1, 0, 1, 1, 1)$ , używając wartości  $k = 6$  i  $n = 4$ . Wyznacz wiadomości, które może podpisać Oskar, znając podpisy pod  $x$  i  $x'$ .
- 6.9. Założmy, że Bolek podpisuje dwa ciągi  $k$ -wyrazowe  $x$  i  $x'$  w schemacie Bosa–Chauma. Niech  $\ell = |\phi(x) \cup \phi(x')|$ . Wykaż, że Oskar może teraz podpisać  $\binom{\ell}{n} - 2$  nowych wiadomości.
- 6.10. Przypuśćmy, że Bolek stosuje schemat podpisu niezaprzeczalnego Chauma–van Antwerpena, taki jak w przykładzie 6.5, a więc z wartościami  $p = 467$ ,  $\alpha = 4$ ,  $a = 101$  oraz  $\beta = 449$ . Założmy, że dociera do niego wiadomość  $x = 157$  z podpisem  $y = 25$  i chce on wykazać, że podpis jest sfałszowany. Niech wybranymi przez Alicję liczbami losowymi do protokołu wyrzeczenia będą:  $e_1 = 46$ ,  $e_2 = 123$ ,  $f_1 = 198$  i  $f_2 = 11$ . Oblicz pytania Alicji,  $c$  i  $d$ , oraz odpowiedzi Bolka,  $C$  i  $D$ , i wykaż, że sprawdzian zgodności przeprowadzony przez Alicję zakończy się powodzeniem.
- 6.11. Wykaż, że każda klasa równoważności kluczów w schemacie podpisu niepodrabialnego Pedersena–van Heysta zawiera dokładnie  $q^2$  kluczów.
- 6.12. Założmy, że Bolek stosuje schemat podpisu niepodrabialnego Pedersena–van Heysta z wartościami  $p = 3467$ ,  $\alpha = 4$ ,  $a_0 = 1567$  oraz  $\beta = 514$  (wartości  $a_0$  Bolek oczywiście nie zna).
- (a) Wiedząc, że  $a_0 = 1567$ , wyznacz wszystkie klucze
- $$K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2),$$
- dla których  $\text{sig}_K(42) = (1118, 1449)$ .
- (b) Przypuśćmy, że  $\text{sig}_K(42) = (1118, 1449)$  oraz  $\text{sig}_K(969) = (899, 471)$ . Nie używając wartości  $a_0 = 1567$ , wyznacz wartość  $K$  (to pokazuje, że schemat jest jednorazowy).

- 6.13. Założymy, że Bolek stosuje schemat podpisu niepodrabialnego Pedersena–van Heysta z wartościami  $p = 5087$ ,  $\alpha = 25$  oraz  $\beta = 1866$ . Niech kluczem będzie

$$K = (5065, 5076, 144, 874, 1873, 2345).$$

Przypuśćmy, że Bolek stwierdził fałszywość podpisu  $(2219, 458)$  pod wiadomością  $4785$ .

- (a) Udowodnij, że sfałszowany podpis spełnia warunek weryfikacji, jest więc ważnym podpisem.
- (b) Opisz, w jaki sposób na podstawie tego podpisu Bolek obliczy dowód fałsterstwa  $a_0$ .

# 7

## Funkcje skrótu

### 7.1. Podpisy i funkcje skrótu

Czytelnik zapewne zauważył, że opisane w rozdziale 6 schematy podpisu są stosowane tylko do „małych” komunikatów. Na przykład, w przypadku użycia DSS podpis pod komunikatem 160-bitowym zajmuje 320 bitów. Na ogół mamy jednak do czynienia ze znacznie dłuższymi tekstami wymagającymi podpisu, tak jak w przypadku dokumentów prawnych. Często zajmują one wiele megalabajtów.

Na pierwszy rzut oka wydawałoby się, że rozwiązanie problemu może polegać na rozbiciu długiego komunikatu na 160-bitowe fragmenty i podpisaniu każdego z nich niezależnie od pozostałych. Przypominałoby to szyfrowanie długiego tekstu jawnego przez niezależne szyfrowanie każdego znaku alfabetu za pomocą tego samego klucza (np. tryb ECB w systemie DES).

Przy takim podejściu wyłania się jednak wiele nowych problemów. Przede wszystkim w przypadku długiej wiadomości otrzymalibyśmy ogromny podpis (w przypadku DSS dwa razy dłuższy od podpisywanego tekstu). Dalej, najbardziej „bezpieczne” schematy podpisu są na ogół powolne, gdyż zazwyczaj stosuje się w nich skomplikowane operacje arytmetyczne, takie jak modularne potęgowanie. Jest jednak jeszcze poważniejszy problem wynikający z tego podejścia: nie można wykluczyć, że niektóre fragmenty podpisanej wiadomości zostaną przestawione, inne usunięte, a mimo to podpis przejdzie pomyślnie procedurę weryfikacji. Musimy zatem chronić nienaruszalność całej wiadomości, czego nie da się osiągnąć, podpisując niezależnie jej małe fragmenty.

Rozwiązaniem wszystkich tych problemów jest zastosowanie bardzo szybkich publicznych *kryptograficznych funkcji skrótu*<sup>†</sup> (ang. *cryptographic hash function*), które z tekstu dowolnej długości tworzą skrót wiadomości (ang. *message digest*) określonej wielkości (160 bitów, gdy używa się DSS). Dopiero do tego skrótu dołącza się podpis. Na rysunku 7.1 przedstawiamy schematycznie użycie funkcji skrótu  $h$  dla DSS.

<sup>†</sup>Są one także zwane funkcjami *haszującymi* lub *ściągającymi* (przyp. tłum.).

wiadomość	$x$	dowolna długość
skrót wiadomości	$z = h(x)$	160 bitów
podpis	$y = \text{sig}_K(z)$	320 bitów

RYSUNEK 7.1. Podpis pod skrótem wiadomości

Gdy Bolek chce podpisać wiadomość  $x$ , buduje najpierw skrót wiadomości  $z = h(x)$ , po czym oblicza podpis  $y = \text{sig}_K(z)$ . Do kanału komunikacyjnego przesyła parę uporządkowaną  $(x, y)$ . Weryfikacja (która może już wykonać każdy) zaczyna się od rekonstrukcji skrótu wiadomości  $z = h(x)$  za pomocą publicznej funkcji skrótu, po której następuje sprawdzenie, czy  $\text{ver}_K(z, y) = \text{tak}$ .

## 7.2. Bezkonfliktowe funkcje skrótu

Stosując funkcję skrótu należy się upewnić, że nie osłabi ona bezpieczeństwa schematu podpisu. Podpisujemy przecież nie wiadomość, a jedynie jej skrót. Trzeba więc narzucić funkcji  $h$  własności, które zapobiegłyby różnego rodzaju fałszerstwom.

Stosując najbardziej oczywisty sposób ataku, przeciwnik, czyli Oskar, zaczyna z ważną podpisaną wiadomością  $(x, y)$ , gdzie  $y = \text{sig}_K(h(x))$  (parę  $(x, y)$ ) może być dowolna wiadomość uprzednio podpisana przez Bolka). Dalej Oskar oblicza  $z = h(x)$  i próbuje znaleźć  $x' \neq x$ , takie że  $h(x') = h(x)$ . Jeżeli mu się to udało, to para  $(x', y)$  byłaby ważną podpisaną wiadomością, a więc fałszerstwem. Aby powstrzymać tego typu ataki, żądamy, aby funkcja  $h$  była bezkonfliktowa (ang. *collision-free*).

### DEFINICJA 7.1

Niech  $x$  będzie dowolną wiadomością. Mówimy, że funkcja skrótu  $h$  jest *słabo bezkonfliktowa* dla  $x$ , gdy znalezienie wiadomości  $x'$ , takiej że  $x' \neq x$  i  $h(x') = h(x)$ , jest obliczeniowo niewykonalne.

Musimy być przygotowani na jeszcze jeden rodzaj ataku. Najpierw Oskar znajduje dwie wiadomości  $x' \neq x$ , takie że  $h(x) = h(x')$ . Następnie przekazuje wiadomość  $x$  Bolkowi i przekonuje go, by podpisał skrót wiadomości  $h(x)$ . W ten sposób uzyskuje  $y$ . Para  $(x', y)$  będzie, jak poprzednio, ważnym fałszerstwem.

Jest to powodem określenia kolejnej własności bezkonfliktowości.

### DEFINICJA 7.2

Funkcja skrótu  $h$  jest *silnie bezkonfliktowa*, jeśli znalezienie wiadomości  $x$  i  $x'$ , takich że  $x \neq x'$  oraz  $h(x) = h(x')$ , jest obliczeniowo niewykonalne.

Zauważmy, że  $h$  jest silnie bezkonfliktowa wtedy i tylko wtedy, gdy obliczeniowo niewykonalne jest znalezienie wiadomości  $x$ , takiej że  $h$  nie jest słabo bezkonfliktowa dla  $x$ .

A oto trzeci wariant ataku. Jak wspomnieliśmy w podrozdziale 6.2, w niektórych schematach podpisu można często sfalszować podpis pod losowym skrótem wiadomości  $z$ . Przypuśćmy, że Oskar oblicza podpis dla takiego losowego skrótu  $z$ , po czym znajduje wiadomość  $x$ , dla której spełniona jest równość  $z = h(x)$ . Gdy mu się to uda, para  $(x, y)$  będzie ważnym fałszerstwem. Aby zapobiec takim atakom, chcemy by funkcja  $h$  była jednokierunkowa w takim sensie, o jakim mówiliśmy przy okazji omawiania systemów kryptograficznych z kluczem publicznym i schematu podpisu Lamporta.

### DEFINICJA 7.3

Funkcja skrótu  $h$  jest *jednokierunkowa*, jeśli dla danego skrótu  $z$  znalezienie wiadomości  $x$ , takiej że  $z = h(x)$ , jest obliczeniowo niewykonalne.

Udowodnimy teraz, że jeśli funkcja jest silnie bezkonfliktowa, to jest także jednokierunkowa. Przeprowadzimy dowód dla implikacji przeciwniej, czyli wykażemy, że każdy algorytm odwracający funkcję skrótu może posłużyć za wyrocznię dla algorytmu probabilistycznego typu Las Vegas wynajdującego konflikty.

Potrzebne są do tego dość słabe założenia o względnych rozmiarach dziedziny i zbioru wartości funkcji skrótu. O funkcji skrótu  $h$  założymy na razie, że  $h : X \rightarrow Z$ , gdzie  $X$  i  $Z$  są zbiorami skończonymi i  $|X| \geq 2|Z|$ . Założenie wydaje się rozsądne: jeśli przyjmujemy, że element zbioru  $X$  jest zakodowany w postaci ciągu bitów o długości  $\log_2 |X|$ , a element zbioru  $Z$  jest zakodowany jako ciąg bitów o długości  $\log_2 |Z|$ , to skrót wiadomości  $z = h(x)$  będzie co najmniej o jeden bit krótszy od samej wiadomości  $x$ . (Później zainteresujemy się przypadkiem, kiedy zbiór możliwych wiadomości  $X$  jest nieskończony, ponieważ chcemy mówić o wiadomościach dowolnej długości; nasze rozumowanie będzie wtedy również słuszne).

Zakładamy, że mamy algorytm odwracający funkcję  $h$ , czyli mamy algorytm **A**, który przyjmuje dane wejściowe w postaci skrótu wiadomości  $z$  i znajduje element  $\mathbf{A}(z) \in X$ , taki że  $h(\mathbf{A}(z)) = z$ .

Udowodnimy następujące twierdzenie.

### TWIERDZENIE 7.1

Niech  $h : X \rightarrow Z$  będzie funkcją skrótu, gdzie  $X$  i  $Z$  są zbiorami skończonymi i  $|X| \geq 2|Z|$ . Założmy, że **A** jest algorytmem odwracającym funkcję  $h$ . Wówczas istnieje algorytm probabilistyczny typu Las Vegas, który dla funkcji  $h$  znajduje konflikt z prawdopodobieństwem co najmniej  $1/2$ .

**DOWÓD** Rozważmy algorytm **B** opisany na rysunku 7.2. Oczywiście **B** jest algorytmem probabilistycznym typu Las Vegas, gdyż albo znajduje konflikt, albo kończy działanie bez odpowiedzi. Nasze główne zadanie polega więc na określeniu

- (1) wybierz losowo  $x \in X$
- (2) oblicz  $z = h(x)$
- (3) oblicz  $x_1 = \mathbf{A}(z)$
- (4) **if**  $x_1 \neq x$  **then**  
 $h$  powoduje konflikt  $x_1$  z  $x$  (sukces)  
**else**  
**QUIT** (porażka)

RYSUNEK 7.2. Użycie odwracającego algorytmu **A** do znajdowania konfliktów dla funkcji skrótu  $h$

prawdopodobieństwa sukcesu. Dla dowolnego  $x \in X$  niech  $x \sim x_1$ , gdy  $h(x) = h(x_1)$ . Łatwo zauważyć, że  $\sim$  jest relacją równoważności. Określmy

$$[x] = \{x_1 \in X : x \sim x_1\}.$$

Każda klasa równoważności  $[x]$  składa się z elementów przeciwbrazu elementu zbioru  $Z$ , zatem liczba takich klas nie przekracza  $|Z|$ . Niech  $C$  oznacza zbiór klas równoważności.

Załóżmy teraz, że  $x$  jest elementem zbioru  $X$  wybranym w kroku 1. Dla tego elementu  $x$  istnieje  $|[x]|$  wartości  $x_1$ , które można otrzymać w kroku 3. Spośród nich  $|[x]| - 1$  elementów jest różnych od  $x$  i dlatego prowadzą one do sukcesu w kroku 4. (Zauważmy, że algorytm **A** nie zna reprezentanta klasy  $[x]$  wybranego w kroku 1). Tak więc prawdopodobieństwo sukcesu przy danym wyborze elementu  $x$  jest równe  $(|[x]| - 1)/|[x]|$ .

Gdy obliczymy średnią ze wszystkich możliwych wyborów elementu  $x$ , otrzymamy prawdopodobieństwo sukcesu algorytmu **B**:

$$\begin{aligned} p(\text{sukces}) &= \frac{1}{|X|} \sum_{x \in X} \frac{|[x]| - 1}{|[x]|} \\ &= \frac{1}{|X|} \sum_{c \in C} \sum_{x \in c} \frac{|c| - 1}{|c|} \\ &= \frac{1}{|X|} \sum_{c \in C} (|c| - 1) \\ &= \frac{1}{|X|} \left( \sum_{c \in C} |c| - \sum_{c \in C} 1 \right) \\ &\equiv \frac{|X| - |Z|}{|X|} \\ &\equiv \frac{|X| - |X|/2}{|X|} \\ &= \frac{1}{2}. \end{aligned}$$

Zbudowaliśmy więc algorytm probabilistyczny typu Las Vegas z prawdopodobieństwem sukcesu nie mniejszym od  $1/2$ .

Wystarczy zatem, by funkcja skrótu była silnie bezkonfliktowa. To pociąga za sobą pozostałe dwie własności. W pozostałej części rozdziału ograniczymy się zatem do rozważania silnie bezkonfliktowych funkcji skrótu.

### 7.3. Atak metodą dnia urodzin

W tym podrozdziale określmy konieczny warunek bezpieczeństwa funkcji skrótu, zależny jedynie od mocy zbioru  $Z$  (lub równoważnie od wielkości skrótu wiadomości). Warunek ten wynika z prostej metody znajdowania konfliktów, znanej nieformalnie pod nazwą *ataku metodą dnia urodzin* (ang. *birthday attack*). Terminologia pochodzi od tzw. *paradoksu dnia urodzin*, który stwierdza, iż w grupie 23 losowo dobranych osób co najmniej dwie mają wspólny dzień urodzin z prawdopodobieństwem równym co najmniej  $1/2$ . (Oczywiście nie jest to w istocie paradoks, lecz fakt przypuszczalnie sprzeczny z intuicją). Powód wyboru takiej nazwy dla metody ataku, o której mowa, stanie się jasny nieco później.

Jak poprzednio założymy, że  $h: X \rightarrow Z$  jest funkcją skrótu,  $X$  i  $Z$  są zbiorami skończonymi oraz  $|X| \geq 2|Z|$ . Niech  $|X| = m$  oraz  $|Z| = n$ . Nietrudno zauważyc, że istnieje co najmniej  $n$  konfliktów, pozostaje jednak pytanie, jak je znaleźć. Można podejść do tego problemu bardzo naiwnie, wybierając losowo  $k$  różnych elementów  $x_1, \dots, x_k \in X$ , obliczając  $z_i = h(x_i)$  i sprawdzając, czy miał miejsce konflikt (np. sortując elementy  $z_i$ ).

Takie postępowanie byłoby analogiczne do losowego wrzucenia  $k$  kulek do  $n$  urn i sprawdzenia, czy w jakiejś urnie znalazły się co najmniej dwie kulki (kulami byłyby losowe wartości  $x_i$ , natomiast urny odpowiadałyby elementom zbioru  $Z$ ).

Obliczymy dolne ograniczenie prawdopodobieństwa wykrycia konfliktu tą metodą. Owo ograniczenie będzie zależało od  $k$  i  $n$ , nie będzie zaś zależało od  $m$ . Ponieważ interesuje nas ograniczenie dolne, przyjmiemy założenie, że  $|h^{-1}(z)| \approx m/n$  dla wszystkich  $z \in Z$ . Założenie wydaje się rozsądne; jeśli przeciobrazy nie są w przybliżeniu równej wielkości, prawdopodobieństwo wykrycia konfliktu wzrasta.

Biorąc więc pod uwagę, że przeciobrazy elementów  $Z$  są (w przybliżeniu) równe, a wartości  $x_i$  wybrane losowo, możemy uznać, iż otrzymane elementy  $z_i$  są także losowymi (niekoniecznie różnymi) elementami zbioru  $Z$ . Nietrudno jednak obliczyć prawdopodobieństwo tego, że  $k$  losowo wybrane elementy są różne. Ustawmy elementy  $z_i$  w kolejności  $z_1, \dots, z_k$ . Wybór pierwszego, czyli  $z_1$ , jest dowolny; prawdopodobieństwo tego, że  $z_2 \neq z_1$  jest równe  $1 - 1/n$ . Z kolei prawdopodobieństwo tego, że  $z_3$  jest różny od  $z_1$  i  $z_2$  jest równe  $1 - 2/n$  itd.

Możemy zatem oszacować prawdopodobieństwo braku konfliktu w sposób następujący:

$$\left(1 - \frac{1}{n}\right) \left(1 - \frac{2}{n}\right) \dots \left(1 - \frac{k-1}{n}\right) = \prod_{i=1}^{k-1} \left(1 - \frac{i}{n}\right).$$

Gdy  $x$  jest rzeczywiście małą liczbą rzeczywistą, mamy przybliżoną równość  $1 - x \approx e^{-x}$ , którą otrzymuje się przez wzięcie dwóch pierwszych wyrazów rozwinięcia

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} \dots$$

W takim razie możemy oszacować prawdopodobieństwo braku konfliktu:

$$\begin{aligned} \prod_{i=1}^{k-1} \left(1 - \frac{i}{n}\right) &\approx \prod_{i=1}^{k-1} e^{-\frac{i}{n}} \\ &= e^{\frac{-k(k-1)}{2n}}. \end{aligned}$$

Stąd wynika, że prawdopodobieństwo zajścia co najmniej jednego konfliktu jest równe

$$1 - e^{\frac{-k(k-1)}{2n}}.$$

Oznaczmy to prawdopodobieństwo symbolem  $\epsilon$ . Wtedy możemy przedstawić  $k$  jako funkcję zmiennych  $n$  i  $\epsilon$ :

$$\begin{aligned} e^{\frac{-k(k-1)}{2n}} &\approx 1 - \epsilon \\ \frac{-k(k-1)}{2n} &\approx \ln(1 - \epsilon) \\ k^2 - k &\approx 2n \ln \frac{1}{1 - \epsilon}. \end{aligned}$$

Pomijając składnik  $-k$ , dochodzimy do oszacowania

$$k \approx \sqrt{2n \ln \frac{1}{1 - \epsilon}}.$$

Dla  $\epsilon = 0,5$  mamy zatem

$$k \approx 1,17\sqrt{n}.$$

Oznacza to, że funkcja skrótu, działając na więcej niż  $\sqrt{n}$  losowych elementach zbioru  $X$ , prowadzi do konfliktu z prawdopodobieństwem  $1/2$ . Zauważmy, że wybór innej wartości  $\epsilon$  daje inny czynnik stały, lecz  $k$  pozostaje proporcjonalne do  $\sqrt{n}$ .

Gdy przyjmiemy, że  $X$  jest zbiorem wszystkich ludzi,  $Y$  zbiorem 365 dni w roku nieprzestępny (a więc bez 29 lutego), a  $h(x)$  oznacza dzień urodzin osoby  $x$ ,

będziemy mieli do czynienia z paradoksem dnia urodzin. Biorąc w naszym oszacowaniu  $n = 365$ , otrzymujemy  $k \approx 22,3$ . Wynika stąd, że jak stwierdziliśmy wcześniej, wśród 23 osób z prawdopodobieństwem co najmniej  $1/2$  znajdą się co najmniej dwie, które obchodzą urodziny tego samego dnia.

Groźba ataku metodą dnia urodzin narzuca dolne ograniczenie długości skrótu wiadomości. Skrót 40-bitowy byłby bardzo niepewny, gdyż wśród nieco ponad  $2^{20}$  (około miliona) losowych skrótów można trafić na konflikt z prawdopodobieństwem  $1/2$ . Zwykle zaleca się uznanie 128 bitów za minimalną dopuszczalną długość skrótu wiadomości (atak metodą dnia urodzin wymagałby wtedy ponad  $2^{64}$  skrótów). Niewątpliwie ustalenie 160-bitowych skrótów w systemie DSS wynikło z tego typu powodów.

#### 7.4. Funkcja skrótu związana z logarytmem dyskretnym

Opiszemy tu wprowadzoną przez Chauma, van Heijsta i Pfitzmannę funkcję skrótu, której bezpieczeństwo wynika z tego, że nie można obliczyć pewnego logarytmu dyskretnego. Funkcja ta jest zbyt wolna, by mogła służyć do celów praktycznych. Jest ona natomiast prosta pojęciowo i stanowi dobry przykład funkcji skrótu, o której można udowodnić, że przy rozsądnych założeniach obliczeniowych jest bezpieczna. **Funkcję skrótu Chauma–van Heijsta–Pfitzmannę** przedstawiamy na rysunku 7.3. Udowodnimy teraz twierdzenie dotyczące jej bezpieczeństwa.

Niech  $p$  będzie dużą liczbą pierwszą, taką że  $q = (p - 1)/2$  jest także liczbą pierwszą. Niech  $\alpha$  i  $\beta$  będą elementami pierwotnymi w  $\mathbb{Z}_p$ . Wartość  $\log_\alpha \beta$  nie jest publiczna, zakładamy również, że obliczenie jej jest obliczeniowo niewykonalne.

Funkcję skrótu

$$h : \{0, \dots, q - 1\} \times \{0, \dots, q - 1\} \rightarrow \mathbb{Z}_p \setminus \{0\}$$

definiujemy następująco:

$$h(x_1, x_2) = \alpha^{x_1} \beta^{x_2} \bmod p.$$

**RYSUNEK 7.3. Funkcja skrótu Chauma–van Heijsta–Pfitzmannę**

#### TWIERDZENIE 7.2

Jeśli funkcja skrótu Chauma–van Heijsta–Pfitzmannę ma tylko jeden konflikt, to można efektywnie obliczyć logarytm dyskretny  $\log_\alpha \beta$ .

**DOWÓD** Założymy, że konflikt ma postać

$$h(x_1, x_2) = h(x_3, x_4)$$

dla par  $(x_1, x_2), (x_3, x_4)$ , takich że  $(x_1, x_2) \neq (x_3, x_4)$ . Mamy zatem następującą kongruencję:

$$\alpha^{x_1} \beta^{x_2} \equiv \alpha^{x_3} \beta^{x_4} \pmod{p},$$

lub

$$\alpha^{x_1 - x_3} \equiv \beta^{x_4 - x_2} \pmod{p}.$$

Niech

$$d = \text{NWD}(x_4 - x_2, p - 1).$$

Wiemy, że  $p - 1 = 2q$  i że  $q$  jest liczbą pierwszą, zatem musi być tak, że  $d \in \{1, 2, q, p - 1\}$ . Rozpatrzmy więc kolejno wszystkie cztery możliwości.

Po pierwsze, przyjmijmy, że  $d = 1$ . Niech

$$y = (x_4 - x_2)^{-1} \pmod{p - 1}.$$

Wtedy

$$\begin{aligned} \beta &\equiv \beta^{(x_4 - x_2)y} \pmod{p} \\ &\equiv \alpha^{(x_1 - x_3)y} \pmod{p}, \end{aligned}$$

możemy więc obliczyć logarytm dyskretny  $\log_{\alpha} \beta$  w sposób następujący:

$$\log_{\alpha} \beta = (x_1 - x_3)(x_4 - x_2)^{-1} \pmod{p - 1}.$$

Założymy teraz, że  $d = 2$ . Wówczas  $\text{NWD}(x_4 - x_2, q) = 1$ , gdyż  $p - 1 = 2q$  i  $q$  jest liczbą nieparzystą. Niech

$$y = (x_4 - x_2)^{-1} \pmod{q}.$$

Dla pewnej liczby całkowitej  $k$  mamy

$$(x_4 - x_2)y = kq + 1,$$

zatem

$$\begin{aligned} \beta^{(x_4 - x_2)y} &\equiv \beta^{kq+1} \pmod{p} \\ &\equiv (-1)^k \beta \pmod{p} \\ &\equiv \pm \beta \pmod{p}, \end{aligned}$$

ponieważ

$$\beta^q \equiv -1 \pmod{p}.$$

Mamy więc

$$\begin{aligned}\beta^{(x_4-x_2)y} &\equiv \alpha^{(x_1-x_3)y} \pmod{p} \\ &\equiv \pm\beta \pmod{p}.\end{aligned}$$

Wynika stąd, że

$$\log_{\alpha} \beta = (x_1 - x_3)y \pmod{p-1}$$

lub

$$\log_{\alpha} \beta = (x_1 - x_3)y + q \pmod{p-1}.$$

Łatwo możemy sprawdzić, która z tych dwóch wersji jest poprawna. Zatem, podobnie jak w przypadku  $d = 1$ , obliczyliśmy logarytm dyskretny  $\log_{\alpha} \beta$ .

Następna możliwość to  $d = q$ . Ale

$$0 \leq x_2 \leq q-1$$

oraz

$$0 \leq x_4 \leq q-1,$$

więc

$$-(q-1) \leq x_4 - x_2 \leq q-1.$$

Tak więc równość  $\text{NWD}(x_4 - x_2, p-1) = q$  nie może mieć miejsca; inaczej mówiąc, ten przypadek nie może się zdarzyć.

Ostatnia możliwość to  $d = p-1$ , co musi oznaczać  $x_4 = x_2$ . Ale wtedy

$$\alpha^{x_1} \beta^{x_2} \equiv \alpha^{x_3} \beta^{x_2} \pmod{p},$$

a więc

$$\alpha^{x_1} \equiv \alpha^{x_3} \pmod{p}$$

i  $x_1 = x_3$ . W rezultacie  $(x_1, x_2) = (x_3, x_4)$  – sprzeczność. Zatem i ten przypadek nie jest możliwy.

Rozważyliśmy wszystkie możliwe wartości  $d$ , możemy więc wnioskować, że funkcja skrótu  $h$  jest bezkonfliktowa, jeśli tylko obliczenie logarytmu dyskretnego  $\log_{\alpha} \beta$  nie jest obliczeniowo wykonalne w  $\mathbb{Z}_p$ . ■

Prześledźmy działanie tego twierdzenia na przykładzie.

### Przykład 7.1

Niech  $p = 12347$  (czyli  $q = 6173$ ),  $\alpha = 2$  oraz  $\beta = 8461$ . Założymy, że mamy konflikt

$$\alpha^{5692} \beta^{144} \equiv \alpha^{212} \beta^{4214} \pmod{12347}.$$

Wówczas  $x_1 = 5692$ ,  $x_2 = 144$ ,  $x_3 = 212$  oraz  $x_4 = 4214$ . W tym przypadku  $\text{NWD}(x_4 - x_2, p - 1) = 2$ , zaczynamy więc od obliczenia

$$\begin{aligned} y &= (x_4 - x_2)^{-1} \bmod q \\ &= (4214 - 144)^{-1} \bmod 6173 \\ &= 4312. \end{aligned}$$

Dalej, obliczamy

$$\begin{aligned} y' &= (x_1 - x_3)y \bmod (p - 1) \\ &= (5692 - 212)4312 \bmod 12346 \\ &= 11862. \end{aligned}$$

Teraz wiemy, że  $\log_\alpha \beta \in \{y', y' + q \bmod (p - 1)\}$ . Z równości

$$\alpha^{y'} \bmod p = 2^{11862} \bmod 12346 = 9998$$

wnioskujemy

$$\begin{aligned} \log_\alpha \beta &= y' + q \bmod (p - 1) \\ &= 11862 + 6173 \bmod 12346 \\ &= 5689. \end{aligned}$$

W celu potwierdzenia wyniku możemy sprawdzić, że

$$2^{5689} \equiv 8461 \pmod{12347}.$$

Obliczyliśmy zatem  $\log_\alpha \beta$ .

□

## 7.5. Rozszerzone funkcje skrótu

Rozważaliśmy dotąd funkcje skrótu określone na skończonej dziedzinie. Teraz zbadamy, jak można rozszerzyć silnie bezkonfliktową funkcję skrótu o skończonej dziedzinie do silnie bezkonfliktowej funkcji skrótu określonej na zbiorze nieskończonym. Pozwoli to na podpisywanie wiadomości dowolnej długości.

Niech  $h: (\mathbb{Z}_2)^m \rightarrow (\mathbb{Z}_2)^t$  będzie silnie bezkonfliktową funkcją skrótu, przy czym  $m \geq t + 1$ . Użyjemy funkcji  $h$  do zbudowania silnie bezkonfliktowej funkcji skrótu  $h^*: X \rightarrow (\mathbb{Z}_2)^t$ , gdzie

$$X = \bigcup_{i=m}^{\infty} (\mathbb{Z}_2)^i.$$

Rozpatrzmy najpierw przypadek, gdy  $m \geq t + 2$ .

Pomyślmy o elementach zbioru  $X$  jako o ciągach bitów. Niech  $|x|$  oznacza długość ciągu  $x$  (czyli liczbę bitów w  $x$ ) i niech  $x||y$  będzie konkatenacją ciągów  $x$  i  $y$ . Założymy, że  $|x| = n > m$ . Ciąg  $x$  możemy przedstawić w postaci konkatenacji

$$x = x_1 \parallel x_2 \parallel \dots \parallel x_k,$$

gdzie

$$|x_1| = |x_2| = \dots = |x_{k-1}| = m - t - 1$$

oraz

$$|x_k| = m - t - 1 - d,$$

gdzie  $0 \leq d \leq m - t - 2$ . Mamy więc

$$k = \left\lceil \frac{n}{m - t - 1} \right\rceil.$$

Funkcję  $h$  określamy za pomocą algorytmu przedstawionego na rysunku 7.4.

- (1) **for**  $i = 1$  **to**  $k - 1$  **do**  
 $y_i = x_i$
  - (2)  $y_k = x_k \parallel 0^d$
  - (3) niech  $y_{k+1}$  będzie binarną reprezentacją  $d$
  - (4)  $g_1 = h(0^{t+1} \parallel y_1)$
  - (5) **for**  $i = 1$  **to**  $k$  **do**  
 $g_{i+1} = h(g_i \parallel 1 \parallel y_{i+1})$
  - (6)  $h^*(x) = g_{k+1}$

RYSUNEK 7.4. Rozszerzenie funkcji skrótu  $h$  do funkcji  $h^*$  ( $m \geq t + 2$ )

Niech

$$y(x) = y_1 \parallel y_2 \parallel \dots \parallel y_{k+1}.$$

Zauważmy, że  $y_k$  powstaje z  $x_k$  przez dopisanie z prawej strony  $d$  zer; dlatego wszystkie bloki  $y_i$  ( $1 \leq i \leq k$ ) mają długość  $m - t - 1$ . Ponadto w kroku 3 element  $y_{k+1}$  powinien być dopełniony z prawej strony zerami, tak by  $|y_{k+1}| = m - t - 1$ .

Proces skracania  $x$  zaczyna się od konstrukcji ciągu  $y(x)$ , po czym bloki  $y_1, y_2, \dots, y_{k+1}$  są w szczególny sposób „przetwarzane”. Ważne jest to, że gdy  $x \neq x'$ , to również  $y \neq y'$ . Definiujemy w istocie  $y_{k+1}$ , tak by przekształcenie  $x \mapsto y(x)$  było różnowartościowe.

Następujące twierdzenie orzeka, że bezpieczeństwo funkcji  $h$  pociąga za sobą bezpieczeństwo funkcji  $h^*$ .

**TWIERDZENIE 7.3**

Niech  $h : (\mathbb{Z}_2)^m \rightarrow (\mathbb{Z}_2)^t$  będzie silnie bezkonfliktową funkcją skrótu, przy czym  $m \geq t + 2$ . Wówczas funkcja  $h^* : \bigcup_{i=m}^{\infty} (\mathbb{Z}_2)^i \rightarrow (\mathbb{Z}_2)^t$ , zbudowana tak jak na rysunku 7.4, jest silnie bezkonfliktową funkcją skrótu.

**DOWÓD** Przypuśćmy, że potrafimy znaleźć elementy  $x \neq x'$ , takie że  $h^*(x) = h^*(x')$ . Pokażemy, w jaki sposób, mając taką parę, można w czasie wielomianowym znaleźć konflikt dla funkcji  $h$ . Otrzymamy w ten sposób sprzeczność z założeniem o bezkonfliktowości tej funkcji, co będzie stanowić dowód, że funkcja  $h^*$  jest silnie bezkonfliktowa.

Niech

$$y(x) = y_1 \| y_2 \| \dots \| y_{k+1}$$

oraz

$$y(x') = y'_1 \| y'_2 \| \dots \| y'_{\ell+1},$$

gdzie  $x$  i  $x'$  są w kroku 2 dopełnione, odpowiednio,  $d$  i  $d'$  zerami. Oznaczmy wartości obliczone w krokach 4 i 5 symbolami  $g_1, \dots, g_{k+1}$  i  $g'_1, \dots, g'_{\ell+1}$ , odpowiednio.

Rozróżnimy dwa przypadki w zależności od tego, czy spełniona jest kongruencja  $|x| \equiv |x'| \pmod{m-t-1}$ .

**Przypadek 1:**  $|x| \not\equiv |x'| \pmod{m-t-1}$ .

W tym przypadku  $d \neq d'$  oraz  $y_{k+1} \neq y'_{\ell+1}$ . Mamy

$$\begin{aligned} h(g_k \| 1 \| y_{k+1}) &= g_{k+1} \\ &= h^*(x) \\ &= h^*(x') \\ &= g'_{\ell+1} \\ &= h(g'_\ell \| 1 \| y'_{\ell+1}), \end{aligned}$$

co daje konflikt dla funkcji  $h$ , gdyż  $y_{k+1} \neq y'_{\ell+1}$ .

**Przypadek 2:**  $|x| \equiv |x'| \pmod{m-t-1}$ .

Dla wygody rozbijemy ten przypadek na dwa podprzypadki.

**Przypadek 2a:**  $|x| = |x'|$ .

Mamy teraz  $k = \ell$  oraz  $y_{k+1} = y'_{\ell+1}$ . Zaczynamy jak w przypadku 1:

$$\begin{aligned} h(g_k \| 1 \| y_{k+1}) &= g_{k+1} \\ &= h^*(x) \\ &= h^*(x') \\ &= g'_{k+1} \\ &= h(g'_k \| 1 \| y'_{k+1}). \end{aligned}$$

Jeśli  $g_k \neq g'_k$ , to mamy już konflikt dla  $h$ , przyjmijmy więc, że  $g_k = g'_k$ . Mamy wówczas

$$\begin{aligned} h(g_{k-1} \parallel 1 \parallel y_k) &= g_k \\ &= g'_k \\ &= h(g'_{k-1} \parallel 1 \parallel y'_k). \end{aligned}$$

Teraz albo mamy konflikt dla  $h$ , albo  $g_{k-1} = g'_{k-1}$  i  $y_k = y'_k$ . Założymy, że konfliktu nie ma; wtedy cofamy się dalej, aż do otrzymania równości

$$\begin{aligned} h(0^{t+1} \parallel y_1) &= g_1 \\ &= g'_1 \\ &= h(0^{t+1} \parallel y'_1). \end{aligned}$$

Jeśli  $y_1 \neq y'_1$ , to mamy konflikt dla  $h$ , więc zakładamy dalej, że  $y_1 = y'_1$ . Ale wówczas  $y_i = y'_i$  dla  $1 \leq i \leq k+1$ , zatem  $y(x) = y(x')$ . To jednak oznacza  $x = x'$ , gdyż przekształcenie  $x \mapsto y(x)$  jest różnowartościowe. Mamy więc sprzeczność z założeniem  $x \neq x'$ .

**Przypadek 2b:**  $|x| \neq |x'|$ .

Bez utraty ogólności możemy założyć, że  $|x'| > |x|$ , więc  $\ell > k$ . Dalej postępujemy podobnie jak w przypadku 2a. Zakładając brak konfliktu dla funkcji  $h$ , dochodzimy ostatecznie do sytuacji, w której

$$\begin{aligned} h(0^{t+1} \parallel y_1) &= g_1 \\ &= g'_{\ell-k+1} \\ &= h(g'_{\ell-k} \parallel 1 \parallel y'_{\ell-k+1}). \end{aligned}$$

Ale w ciągu  $0^{t+1} \parallel y_1$  bitem o numerze  $t+1$  jest 0, podczas gdy w  $g'_{\ell-k} \parallel 1 \parallel y'_{\ell-k+1}$  na tym miejscu znajduje się 1. W ten sposób otrzymaliśmy konflikt dla  $h$ .

Rozpatrzyliśmy wszystkie możliwe przypadki, otrzymaliśmy zatem żądaną tezę. ■

Konstrukcję z rysunku 7.4 można stosować tylko wtedy, gdy  $m \geq t+2$ . Zobaczmy teraz, co się dzieje, gdy  $m = t+1$ . Musimy w tej sytuacji szukać innej drogi do zbudowania funkcji  $h^*$ . Założymy, jak poprzednio, że  $|x| = n > m$ . Najpierw zakodujemy w szczególny sposób ciąg  $x$ . Użyjemy do tego funkcji  $f$  zdefiniowanej jak następuje:

$$\begin{aligned} f(0) &= 0 \\ f(1) &= 01. \end{aligned}$$

Algorytm konstrukcji funkcji  $h^*$  przedstawiamy na rysunku 7.5.

- (1) niech  $y = y_1y_2 \dots y_k = 11 \parallel f(x_1) \parallel f(x_2) \parallel \dots \parallel f(x_n)$
- (2)  $g_1 = h(0^t \parallel y_1)$
- (3) **for**  $i = 1$  **to**  $k - 1$  **do**  

$$g_{k+1} = h(g_i \parallel y_{i+1})$$
- (4)  $h^*(x) = g_k$

**RYSUNEK 7.5.** Rozszerzenie funkcji skrótu  $h$  do funkcji  $h^*$  ( $m = t + 1$ )

Kodowanie  $x \mapsto y = y(x)$ , określone w kroku 1, ma dwie zasadnicze właściwości:

- (1) Jeśli  $x \neq x'$ , to  $y(x) \neq y(x')$  (co oznacza, że przyporządkowanie  $x \mapsto y(x)$  jest różnowartościowe).
- (2) Nie istnieją dwa ciągi  $x \neq x'$  i ciąg  $z$ , takie że  $y(x) = z \parallel y(x')$ . (Innymi słowy, żadne kodowanie nie jest *przyrostkiem* innego kodowania. Łatwo się o tym przekonać, bowiem każdy ciąg  $y(x)$  zaczyna się od 11, natomiast nigdy w dalszej części ciągu nie występują obok siebie dwie jedynki).

#### TWIERDZENIE 7.4

Niech  $h : (\mathbb{Z}_2)^{t+1} \rightarrow (\mathbb{Z}_2)^t$  będzie silnie bezkonfliktową funkcją skrótu. Wówczas funkcja  $h^* : \bigcup_{i=t+1}^{\infty} (\mathbb{Z}_2)^i \rightarrow (\mathbb{Z}_2)^t$ , określona tak jak na rysunku 7.5, jest również silnie bezkonfliktową funkcją skrótu.

**DOWÓD** Przypuśćmy, że istnieją  $x \neq x'$ , takie że  $h^*(x) = h^*(x')$ . Niech

$$y(x) = y_1y_2 \dots y_k$$

oraz

$$y(x') = y'_1y'_2 \dots y'_{\ell}.$$

Rozpatrzmy dwa przypadki.

**Przypadek 1:**  $k = \ell$ .

Jak w twierdzeniu 7.3, albo znajdujemy konflikt dla  $h$ , albo dochodzimy do  $y = y'$ . Z tej równości wynika jednak  $x = x'$  – sprzeczność.

**Przypadek 2:**  $k \neq \ell$ .

Bez utraty ogólności przyjmijmy  $\ell > k$ . W tym przypadku postępujemy podobnie jak poprzednio. Zakładając brak konfliktu dla  $h$ , dochodzimy do ciągu równości

$$\begin{aligned} y_k &= y'_{\ell} \\ y_{k-1} &= y'_{\ell-1} \\ &\vdots && \vdots \\ y_1 &= y'_{\ell-k+1}, \end{aligned}$$

co przeczy jednak własności dotyczącej przyrostków, którą sformułowaliśmy wyżej.

Wnioskujemy zatem, że funkcja  $h^*$  jest bezkonfliktowa. ■

Podsumujmy obie konstrukcje funkcji  $h^*$  i ustalmy, ile razy trzeba użyć funkcji  $h$  do obliczenia jej rozszerzenia.

### TWIERDZENIE 7.5

Niech  $h : (\mathbb{Z}_2)^m \rightarrow (\mathbb{Z}_2)^t$  będzie silnie bezkonfliktową funkcją skrótu, przy czym  $m \geq t + 1$ . Wówczas istnieje silnie bezkonfliktowa funkcja skrótu

$$h^* : \bigcup_{i=m}^{\infty} (\mathbb{Z}_2)^i \rightarrow (\mathbb{Z}_2)^t.$$

Podczas obliczania  $h^*(x)$  trzeba obliczyć wartość funkcji  $h$  co najwyżej  $k$  razy, gdzie

$$k = \begin{cases} 1 + \left\lceil \frac{n}{m-t-1} \right\rceil, & \text{gdy } m \geq t+2 \\ 2n+2, & \text{gdy } m = t+1, \end{cases}$$

gdzie  $|x| = n$ .

## 7.6. Funkcje skrótu budowane na systemach kryptograficznych

Opisane dotąd metody dają w wyniku funkcje skrótu zbyt wolne, by były użyteczne w praktycznych zastosowaniach. Można jednak do konstrukcji funkcji skrótu wykorzystać istniejący system kryptograficzny z kluczem prywatnym. Niech  $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$  będzie obliczeniowo bezpiecznym systemem kryptograficznym. Dla wygody przyjmijmy, że  $\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbb{Z}_2)^n$ . Powinniśmy tu założyć dodatkowo, że  $n \geq 128$ , choćby po to, by zapobiec atakowi metodą dnia urodzin. Wyklucza to z naszych rozważań system DES (podobnie jak i fakt, że długość klucza w DES jest różna od długości tekstu jawnego).

Załóżmy, że mamy ciąg bitów

$$x = x_1 \parallel x_2 \parallel \dots \parallel x_k,$$

gdzie  $x_i \in (\mathbb{Z}_2)^n$ ,  $1 \leq i \leq k$ . (Gdyby liczba bitów ciągu  $x$  nie była wielokrotnością  $n$ , należałoby  $x$  w jakiś sposób dopełnić, tak jak to zrobiliśmy w podrozdziale 7.5. Dla uproszczenia pominimy tu ten aspekt).

Zasadniczy pomysł polega na tym, że zaczynamy od pewnej „wartości początkowej”  $WP = g_0$ , po czym obliczamy kolejno  $g_1, \dots, g_k$  za pomocą reguły postaci

$$g_i = f(x_i, g_{i-1}),$$

gdzie  $f$  jest funkcją opartą na funkcji szyfrowania wybranego systemu kryptograficznego. Na koniec definiujemy skrót wiadomości  $h(x) = g_k$ .

W literaturze pojawiło się wiele propozycji funkcji skrótu tego typu, jednak dla wielu z nich wykazano, że nie zapewniają bezpieczeństwa (niezależnie od bezpieczeństwa zastosowanego systemu kryptograficznego). Cztery warianty wydają się jednak bezpieczne:

$$\begin{aligned}g_i &= e_{g_{i-1}}(x_i) \oplus x_i \\g_i &= e_{g_{i-1}}(x_i) \oplus x_i \oplus g_{i-1} \\g_i &= e_{g_{i-1}}(x_i \oplus g_{i-1}) \oplus x_i \\g_i &= e_{g_{i-1}}(x_i \oplus g_{i-1}) \oplus x_i \oplus g_{i-1}.\end{aligned}$$


---

## 7.7. Funkcja skrótu MD4

Funkcję skrótu MD4 zaproponował Rivest w 1990 roku; w roku 1991 pojawiła się wzmacniona wersja, nazwana MD5. **Bezpieczny standard skrótu** (zwany w skrócie SHS od ang. *Secure Hash Standard*) jest bardziej złożony, choć oparty na tych samych metodach. Opublikowano go w Rejestrze Federalnym dnia 31 stycznia 1992 roku, a za standard uznano 11 maja 1993 roku. (Zaproponowaną wówczas poprawkę, usuwającą „techniczną lukę” w SHS, wprowadzono 11 lipca 1994 roku). Wszystkie wymienione tu funkcje skrótu są bardzo szybkie, a więc praktycznie użyteczne do podpisywania bardzo długich wiadomości.

Omówimy tu szczegółowo MD4, poświęcając nieco miejsca pewnym modyfikacjom zastosowanym w MD5 i SHS.

Dla danego ciągu bitów  $x$  tworzymy najpierw macierz

$$M = M[0]M[1]\dots M[N-1],$$

w której  $M[i]$  są ciągami 32-bitowymi, a  $N \equiv 0 \pmod{16}$ . Każdy blok  $M[i]$  nazywiemy *slowem*. Algorytm, przedstawiony na rysunku 7.6, buduje macierz  $M$  z ciągu  $x$ .

- (1)  $d = (447 - |x|) \pmod{512}$
- (2) niech  $\ell$  będzie binarną reprezentacją liczby  $|x| \pmod{2^{64}}$ ,  $|\ell| = 64$
- (3)  $M = x \parallel 1 \parallel 0^d \parallel \ell$

**RYSUNEK 7.6. Konstrukcja macierzy  $M$  w MD4**

Konstruując macierz  $M$ , dodajemy do  $x$  jeden bit 1. Następnie dopełniamy zerami, tak by długość otrzymanego ciągu przystawała do liczby 448 modulo 512,

i na końcu dołączamy 64 bity składające się na binarną reprezentację (pierwotnej) długości ciągu  $x$  (zredukowanej modulo 64, jeśli to konieczne). Długość wynikowego ciągu  $M$  jest podzielna przez 512, stąd, jeśli rozbijemy  $M$  na słowa 32-bitowe, ich liczba, oznaczana symbolem  $N$ , jest podzielna przez 16.

Przystąpimy teraz do skonstruowania 128-bitowego skrótu wiadomości. Na rysunku 7.7 znajduje się ogólny opis algorytmu. Skrót wiadomości powstaje jako konkatenacja czterech słów  $A$ ,  $B$ ,  $C$  i  $D$ , które nazwiemy *rejestrami*. W kroku 1 określamy wartości początkowe czterech rejestrów. Następnie przetwarzamy macierz  $M$  w pakietach po 16 słów. W kroku 2, w każdej iteracji pętli, pobieramy najpierw kolejnych 16 słów macierzy  $M$  i zapisujemy je w macierzy  $X$  (krok 3).

- ```

(1)  $A = 67452301$  (heks)
      $B = efcdab89$  (heks)
      $C = 98badcfe$  (heks)
      $D = 10325476$  (heks)
(2) for  $i = 0$  to  $N/16 - 1$  do
(3)   for  $j = 0$  to 15 do
         $X[j] = M[16i + j]$ 
(4)    $AA = A$ 
        $BB = B$ 
        $CC = C$ 
        $DD = D$ 
(5)   Runda1
(6)   Runda2
(7)   Runda3
(8)    $A = A + AA$ 
        $B = B + BB$ 
        $C = C + CC$ 
        $D = D + DD$ 

```

**RYSUNEK 7.7.** Funkcja skrótu MD4

Następnie zapisujemy wartości czterech rejestrów (krok 4). Potem wykonujemy trzy rundy skracania. Każda runda składa się z jednej operacji na każdym z 16 słów w  $X$  (nieco dalej omówimy te operacje bardziej szczegółowo); operacje wykonane w trzech rundach prowadzą do nowych wartości w rejestrach. Wreszcie w kroku 8 uaktualniamy wartości rejestrów przez ponowne dodanie wartości zapisanych w kroku 4, przy czym chodzi tu o dodawanie dodatnich liczb całkowitych modulo  $2^{32}$ .

Trzy rundy w MD4 różnią się między sobą (w odróżnieniu od DES, gdzie mamy 16 identycznych rund). Najpierw opiszemy kilka operacji stosowanych w tych rundach. W tym opisie  $X$  i  $Y$  oznaczają słowa wejściowe, a wynikiem

każdej operacji jest słowo wyjściowe. Oto te operacje (z których pierwsze cztery są wykonane po współrzędnych, czyli na poziomie bitów):

|              |                                                                        |
|--------------|------------------------------------------------------------------------|
| $X \wedge Y$ | koniunkcja $X$ i $Y$                                                   |
| $X \vee Y$   | alternatywa $X$ i $Y$                                                  |
| $X \oplus Y$ | różnica symetryczna $X$ i $Y$                                          |
| $\neg X$     | dopełnienie $X$                                                        |
| $X + Y$      | dodawanie liczb całkowitych modulo $2^{32}$                            |
| $X \lll s$   | cykliczne przesunięcie $X$ o $s$ pozycji w lewo ( $0 \leq s \leq 31$ ) |

Wszystkie te operacje są bardzo szybkie, jedyną operacją arytmetyczną jest dodawanie modulo  $2^{32}$ . W praktyce trzeba jeszcze uwzględnić architekturę komputera, na którym MD4 jest implementowana, aby dodawanie było wykonywane poprawnie. Założmy, że słowo składa się z czterech bajtów  $a_1a_2a_3a_4$ . Każdy z nich możemy interpretować jako binarną reprezentację liczby całkowitej z zakresu od 0 do 255. W architekturze „najpierw najstarszy bit” (ang. *big-endian*) (jak w przypadku komputera Sun SPARCstation) takie słowo reprezentuje liczbę całkowitą

$$a_12^{24} + a_22^{16} + a_32^8 + a_4,$$

w innych, o architekturze „najpierw najmłodszy bit” (ang. *little endian*) (wykorzystujących procesory serii Intel 80xxx), reprezentuje ono liczbę

$$a_42^{24} + a_32^{16} + a_22^8 + a_1.$$

W konstrukcji MD4 zakłada się to drugie rozwiązanie. Ważne jest jednak to, że skrót wiadomości nie zależy od rzeczywistej architektury komputera. Jeśli więc chcemy realizować MD4 na komputerze o architekturze „najpierw najstarszy bit”, musimy wykonywać dodawanie w sposób następujący:

- (1) Zamienić  $x_1$  z  $x_4$ ,  $x_2$  z  $x_3$ ,  $y_1$  z  $y_4$  oraz  $y_2$  z  $y_3$ .
- (2) Obliczyć  $Z = X + Y$  modulo  $2^{32}$ .
- (3) Zamienić  $z_1$  z  $z_4$  oraz  $z_2$  z  $z_3$ .

W trzech rundach MD4 stosowane są trzy funkcje:  $f$ ,  $g$  oraz  $h$ . Każda z nich jest funkcją boolowską wykonywaną na bitach, każda z nich przyjmuje na wejściu trzy słowa i daje w wyniku jedno słowo. Oto ich definicje:

$$\begin{aligned} f(X, Y, Z) &= (X \wedge Y) \vee ((\neg X) \wedge Z) \\ g(X, Y, Z) &= (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z) \\ h(X, Y, Z) &= X \oplus Y \oplus Z. \end{aligned}$$

Pełny opis wszystkich trzech rund funkcji MD4 przedstawiamy na rysunkach 7.8–7.10.

MD4 zaprojektowano tak, aby jej działanie było bardzo szybkie – i rzeczywiście, implementacje programowe na Sun SPARCstation osiągają szybkość 1,4 MB

- (1)  $A = (A + f(B, C, D) + X[0]) \lll 3$
- (2)  $D = (D + f(A, B, C) + X[1]) \lll 7$
- (3)  $C = (C + f(D, A, B) + X[2]) \lll 11$
- (4)  $B = (B + f(C, D, A) + X[3]) \lll 19$
- (5)  $A = (A + f(B, C, D) + X[4]) \lll 3$
- (6)  $D = (D + f(A, B, C) + X[5]) \lll 7$
- (7)  $C = (C + f(D, A, B) + X[6]) \lll 11$
- (8)  $B = (B + f(C, D, A) + X[7]) \lll 19$
- (9)  $A = (A + f(B, C, D) + X[8]) \lll 3$
- (10)  $D = (D + f(A, B, C) + X[9]) \lll 7$
- (11)  $C = (C + f(D, A, B) + X[10]) \lll 11$
- (12)  $B = (B + f(C, D, A) + X[11]) \lll 19$
- (13)  $A = (A + f(B, C, D) + X[12]) \lll 3$
- (14)  $D = (D + f(A, B, C) + X[13]) \lll 7$
- (15)  $C = (C + f(D, A, B) + X[14]) \lll 11$
- (16)  $B = (B + f(C, D, A) + X[15]) \lll 19$

**RYSUNEK 7.8.** Runda 1 MD4

na sekundę. Jednakże trudno coś orzec o bezpieczeństwie funkcji skrótu typu MD4, ponieważ nie jest to funkcja oparta na dobrze zbadanym problemie, takim jak faktoryzacja czy problem logarytmu dyskretnego. Stąd, podobnie jak w przypadku systemu DES, ufność w bezpieczeństwo systemu zapewni dopiero upływ czasu, jeśli poświęci mu się więcej badań i (miejmy nadzieję) nie okaże się on niepewny.

Choć system MD4 nie został dotąd złamany, bez szczególnego trudu można złamać jego słabsze wersje, w których pomija się pierwszą lub trzecią rundę. Łatwo można bowiem znaleźć konflikt dla tych dwurundowych wariantów. W 1991 roku pojawiła się wersja wzmacniona, nazwana MD5, w której stosuje się cztery rundy zamiast trzech – kosztem utraty około 30% szybkości w stosunku do MD4 (o ok. 0,9 MB/s na SPARCstation).

Bezpieczny standard skrótu jest jeszcze bardziej skomplikowany i jeszcze wolniejszy (o ok. 0,2 MB/s na SPARCstation). Nie podamy tu pełnego opisu, przedstawimy jedynie niektóre modyfikacje zastosowane w SHS.

- (1) SHS został zaprojektowany z myślą raczej o architekturze „najpierw najstarszy bit”, a nie o architekturze „najpierw najmłodszy bit”.
- (2) SHS tworzy 5-rejestrowy (160-bitowy) skrót wiadomości.
- (3) SHS przetwarza jednocześnie 16 słów, podobnie jak MD4, jednakże 16 słów jest najpierw „rozszerzanych” do 80 słów. Następnie wykonuje się ciąg 80 operacji, po jednej na każdym słowie.

- (1)  $A = (A + g(B, C, D) + X[0]) + 5A827999) \lll 3$
- (2)  $D = (D + g(A, B, C) + X[4]) + 5A827999) \lll 5$
- (3)  $C = (C + g(D, A, B) + X[8]) + 5A827999) \lll 9$
- (4)  $B = (B + g(C, D, A) + X[12]) + 5A827999) \lll 13$
- (5)  $A = (A + g(B, C, D) + X[1]) + 5A827999) \lll 3$
- (6)  $D = (D + g(A, B, C) + X[5]) + 5A827999) \lll 5$
- (7)  $C = (C + g(D, A, B) + X[9]) + 5A827999) \lll 9$
- (8)  $B = (B + g(C, D, A) + X[13]) + 5A827999) \lll 13$
- (9)  $A = (A + g(B, C, D) + X[2]) + 5A827999) \lll 3$
- (10)  $D = (D + g(A, B, C) + X[6]) + 5A827999) \lll 5$
- (11)  $C = (C + g(D, A, B) + X[10]) + 5A827999) \lll 9$
- (12)  $B = (B + g(C, D, A) + X[14]) + 5A827999) \lll 13$
- (13)  $A = (A + g(B, C, D) + X[3]) + 5A827999) \lll 3$
- (14)  $D = (D + g(A, B, C) + X[7]) + 5A827999) \lll 5$
- (15)  $C = (C + g(D, A, B) + X[11]) + 5A827999) \lll 9$
- (16)  $B = (B + g(C, D, A) + X[15]) + 5A827999) \lll 13$

**RYSUNEK 7.9. Runda 2 MD4**

A oto jak działa funkcja „rozszerzania”. Dla danych 16 słów  $X[0], \dots, X[15]$  obliczamy 64 dalsze słowa zgodnie z zależnością rekurencyjną

$$X[j] = X[j - 3] \oplus X[j - 8] \oplus X[j - 14] \oplus X[j - 16], \quad 16 \leq j \leq 79. \quad (7.1)$$

W wyniku wykonania obliczeń opisanych równaniem (7.1) każde ze słów  $X[16], \dots, X[79]$  powstaje jako różnica symetryczna określonego podzbioru słów  $X[0], \dots, X[15]$ .

Na przykład, mamy

$$X[16] = X[0] \oplus X[2] \oplus X[8] \oplus X[13]$$

$$X[17] = X[1] \oplus X[3] \oplus X[9] \oplus X[14]$$

$$X[18] = X[2] \oplus X[4] \oplus X[10] \oplus X[15]$$

$$X[19] = X[0] \oplus X[2] \oplus X[3] \oplus X[5] \oplus X[8] \oplus X[11] \oplus X[13]$$

 $\vdots$ 

$$X[79] = X[1] \oplus X[4] \oplus X[5] \oplus X[8] \oplus X[9] \oplus X[12] \oplus X[13]$$

Zaproponowane poprawki dotyczą funkcji rozszerzania. Proponuje się mianowicie, by równanie (7.1) zastąpić następującym równaniem:

$$X[j] = (X[j - 3] \oplus X[j - 8] \oplus X[j - 14] \oplus X[j - 16]) \lll 1, \quad 16 \leq j \leq 79. \quad (7.2)$$

Jak poprzednio, operacja „ $\lll 1$ ” oznacza cykliczne przesunięcie w lewo o jedną pozycję.

- (1)  $A = (A + h(B, C, D) + X[0]) + 6ED9EBA1) \lll 3$
- (2)  $D = (D + h(A, B, C) + X[8]) + 6ED9EBA1) \lll 9$
- (3)  $C = (C + h(D, A, B) + X[4]) + 6ED9EBA1) \lll 11$
- (4)  $B = (B + h(C, D, A) + X[12]) + 6ED9EBA1) \lll 15$
- (5)  $A = (A + h(B, C, D) + X[2]) + 6ED9EBA1) \lll 3$
- (6)  $D = (D + h(A, B, C) + X[10]) + 6ED9EBA1) \lll 9$
- (7)  $C = (C + h(D, A, B) + X[6]) + 6ED9EBA1) \lll 11$
- (8)  $B = (B + h(C, D, A) + X[14]) + 6ED9EBA1) \lll 15$
- (9)  $A = (A + h(B, C, D) + X[1]) + 6ED9EBA1) \lll 3$
- (10)  $D = (D + h(A, B, C) + X[9]) + 6ED9EBA1) \lll 9$
- (11)  $C = (C + h(D, A, B) + X[5]) + 6ED9EBA1) \lll 11$
- (12)  $B = (B + h(C, D, A) + X[13]) + 6ED9EBA1) \lll 15$
- (13)  $A = (A + h(B, C, D) + X[3]) + 6ED9EBA1) \lll 3$
- (14)  $D = (D + h(A, B, C) + X[11]) + 6ED9EBA1) \lll 9$
- (15)  $C = (C + h(D, A, B) + X[7]) + 6ED9EBA1) \lll 11$
- (16)  $B = (B + h(C, D, A) + X[15]) + 6ED9EBA1) \lll 15$

RYSUNEK 7.10. Runda 3 MD4

## 7.8. Datowanie

Jednym z problemów związanych ze schematami podpisu jest to, że algorytm podpisu może zostać ujawniony. Przypuśćmy, na przykład, że Oskar jest w stanie ustalić tajny wykładowik  $a$ , którym posługuje się Bolek w systemie DSS. Wówczas, oczywiście, Oskar może fałszować podpis Bolka pod dowolną wiadomością. Pojawia się tu jeszcze inna trudność (zapewne jeszcze poważniejsza): kompromitacja algorytmu podpisu podważa wiarygodność wszystkich wiadomości podpisanych przez Bolka, łącznie z tymi, które podpisał przed przechwytem algorytmu przez Oskara.

Wyobraźmy sobie jeszcze inną, nieprzyjemną sytuację. Otóż Bolek może podpisać wiadomość, a następnie wyprzeć się jej. Wystarczy, że ujawni swój algorytm podpisu, by potem twierdzić, że podpis pod wiadomością jest fałszerstwem.

Takie sytuacje są możliwe dlatego, że nie ma sposobu ustalenia, kiedy podpis pod wiadomością został złożony. Sugeruje to konieczność rozważenia metody opatrzenia datą i czasem (podpisanej) wiadomości za pomocą *datownika*<sup>†</sup> (ang. *time-stamp*). Datownik stanowiłby dowód, że wiadomość została podpisana w określonym czasie. Wówczas, nawet gdyby algorytm Bolka uległ kompromitacji, nie unieważniałoby to wszystkich jego wcześniejszych podpisów. Podobnie

<sup>†</sup>W literaturze funkcjonuje również termin *znacznik czasu* (przyp. tłum.).

funkcjonują karty kredytowe: gdy ktoś zgubi taką kartę i zawiadomi o tym bank, karta ulega unieważnieniu, ale nie wpływa to na zakupy poczynione przed jej zagubieniem.

Opiszymy tu kilka metod datowania. Po pierwsze, zauważmy, że Bolek może sam utworzyć przekonujący datownik. Najpierw wchodzi w posiadanie aktualnej, publicznie dostępnej i nie dającej się wcześniej przewidzieć informacji. Mogłyby to być, na przykład, wyniki najważniejszych rozgrywek baseballowych z poprzedniego dnia albo wartości wszystkich akcji, którymi obraca nowojorska giełda. Niech  $pub$  oznacza taką informację.

Załóżmy teraz, że Bolek chce opatrzyć podpis pod wiadomością  $x$  datownikiem. Przyjmijmy, że  $h$  jest publicznie znaną funkcją skrótu. Bolek postępuje zgodnie z algorytmem przedstawionym na rysunku 7.11. Oto jak ten schemat działa. Po pierwsze, wystąpienie informacji  $pub$  oznacza, że Bolek nie mógł wygenerować podpisu  $y$  wcześniej niż to określa datownik. Po drugie, opublikowanie  $y$  następnego dnia w prasie dowodzi, że Bolek nie obliczył  $y$  po tej dacie. W rezultacie podpis Bolka jest ograniczony do jednego dnia. Zauważmy też, że Bolek nie ujawnia w tym schemacie samej wiadomości  $x$ , gdyż publikacji podlega tylko  $z$ . W razie potrzeby Bolek może udowodnić, że podpisana i datowaną przez niego wiadomością była wiadomość  $x$ , po prostu ujawniając ją.

- (1) Bolek oblicza  $z = h(x)$
- (2) Bolek oblicza  $z' = h(z \parallel pub)$
- (3) Bolek oblicza  $y = sig_K(z')$
- (4) Bolek publikuje  $(z, pub, y)$  w gazecie na następny dzień

**RYSUNEK 7.11. Datowanie podpisu pod wiadomością  $x$**

Bez trudu można też tworzyć datowniki, jeśli istnieje dostępna i wiarygodna usługa datowania (np. elektroniczny notariusz). Bolek może wtedy obliczyć  $z = h(x)$  oraz  $y = sig_K(z)$ , a następnie wysłać  $(z, y)$  do usługi datowania (ang. *timestamping service*, TSS). TSS opatrzy datownikiem  $D$  i podpisze trójkę  $(z, y, D)$ . Taki system świetnie działa, jednak pod warunkiem, że algorytm podpisu TSS pozostaje bezpieczny, a TSS nie jest podatny na przekupstwo zmierzające do antydatowania podpisu. Zauważmy jeszcze, że opisana tu metoda pozwala jedynie ustalić, że Bolek podpisał wiadomość przed określona datą. Gdyby chciał on wykazać, że nastąpiło to po pewnym dniu, mógłby użyć publicznej informacji  $pub$ , tak jak w poprzedniej metodzie.

Jeśli bezwarunkowe zaufanie do TSS nie jest wskazane, można zwiększyć bezpieczeństwo przez sekwencyjne łączenie datowanych wiadomości. Zgodnie z takim schematem Bolek wysyła do TSS uporządkowaną trójkę  $(z, y, ID(Bolek))$ , w której  $z$  jest skrótem wiadomości  $x$ ,  $y$  jest podpisem Bolka pod skrótem  $z$ , natomiast  $ID(Bolek)$  jest informacją identyfikującą Bolka. TSS będzie datował ciąg tego typu trójkę. Niech  $(z_n, y_n, ID_n)$  oznacza  $n$ -tą trójkę podlegającą datowaniu przez TSS i niech  $t_n$  określa czas, w którym składane jest  $n$ -te zamówienie.

TSS datuje wtedy  $n$ -tą trójkę za pomocą algorytmu przedstawionego na rysunku 7.12. Wielkość  $L_n$  jest „informacją wiążącą”, łączącą  $n$ -te zamówienie z poprzednim ( $L_0$  jest wtedy dowolną określona z góry nieistotną informacją, potrzebną do zainicjowania procesu).

- (1) TSS oblicza  $L_n = (t_{n-1}, \text{ID}_{n-1}, z_{n-1}, y_{n-1}, h(L_{n-1}))$
  - (2) TSS oblicza  $C_n = (n, t_n, z_n, y_n, \text{ID}_n, L_n)$
  - (3) TSS oblicza  $s_n = \text{sig}_{\text{TSS}}(h(C_n))$
  - (4) TSS przesyła  $(C_n, s_n, \text{ID}_{n+1})$  do  $\text{ID}_n$

**RYSUNEK 7.12. Datowanie ( $z_n, y_n, \text{ID}_n$ )**

Na żądanie Bolek może teraz ujawnić swoją wiadomość  $x_n$ , umożliwiając weryfikację podpisu  $y_n$ . Następnie można zweryfikować podpis  $s_n$  dodany przez TSS. Jeśli trzeba, można dalej zażądać datowników dla  $\text{ID}_{n-1}$  lub  $\text{ID}_{n+1}$ , czyli  $(C_{n-1}, s_{n-1}, \text{ID}_n)$  i  $(C_{n+1}, s_{n+1}, \text{ID}_{n+2})$ , odpowiednio, i w tych datownikach sprawdzić podpisy TSS. Oczywiście ten proces można powtarzać dowolnie długo, przesuwając się naprzód lub do tyłu.

## 7.9. Uwagi i bibliografia

Funkcja skrótu oparta na logarytmie dyskretnym, opisana w podrozdziale 7.4, pochodzi od Chauma, van Heijsta i Pfitzmann [CvHP92]. Gibson [GIB91] podał funkcję skrótu, której bezpieczeństwo zależy od niewykonalności rozkładu na czynniki pierwsze pewnej liczby złożonej  $n$  (ćwiczenie 7.4 zawiera opis tego schematu).

Omówienie rozszerzonej funkcji skrótu w podrozdziale 7.5 jest oparte na pracy Damgård'a [DA90]. Podobne metody odkrył Merkle [ME90].

Więcej informacji o funkcjach skrótu wykorzystujących systemy kryptograficzne z kluczem prywatnym można znaleźć w artykule Preneela, Govaertsa i Vandewalle'a [PGV94].

Algorytm skrótu MD4 przedstawił Rivest [Ri91], natomiast bezpieczny standard skrótu jest opisany w [NBS93]. W artykule [pBB92] den Boer i Bossalaers podają sposób ataku na dwie z trzech rund systemu MD4. Wśród ostatnio propozowanych funkcji skrótu są jeszcze funkcje **N-hash** [MOI90] oraz **Snefru** [ME90A].

Datowania dotyczą też artykuły Habera i Stornetty [HS91] oraz Bayera, Habera i Stornetty [BHS93].

Preneel, Govaerts i Vandewalle przedstawiają w [PGV93] szeroki przegląd technik skrótu.

## Ćwiczenia

7.1. Niech  $h : X \rightarrow Y$  będzie funkcją skrótu. Dla każdego  $y \in Y$  niech

$$h^{-1}(y) = \{x : h(x) = y\}$$

i  $s_y = |h^{-1}(y)|$ . Zdefiniujmy

$$N = |\{\{x_1, x_2\} : h(x_1) = h(x_2)\}|.$$

Liczba  $N$  określa liczbę nieuporządkowanych par w  $X$ , które powodują konflikt dla  $h$ .

(a) Wykaż, że

$$\sum_{y \in Y} s_y = |X|,$$

zatem średnią dla wszystkich  $s_y$  jest

$$\bar{s} = \frac{|X|}{|Y|}.$$

(b) Wykaż, że

$$N = \sum_{y \in Y} \binom{s_y}{2} = \frac{1}{2} \sum_{y \in Y} s_y^2 - c \frac{|X|}{2}.$$

(c) Wykaż, że

$$\sum_{y \in Y} (s_y - \bar{s})^2 = 2N + |X| - \frac{|X|^2}{|Y|}.$$

(d) Korzystając z wyniku udowodnionego w (c), wykaż że

$$N \geq \frac{1}{2} \left( \frac{|X|^2}{|Y|} - |X| \right),$$

a równość zachodzi wtedy i tylko wtedy, gdy

$$s_y = \frac{|X|}{|Y|}$$

dla każdego  $y \in Y$ .

7.2. Założmy, tak jak w ćwiczeniu 7.1, że  $h : X \rightarrow Y$  jest funkcją skrótu i niech

$$h^{-1}(y) = \{x : h(x) = y\}$$

dla dowolnego  $y \in Y$ . Niech  $\epsilon$  oznacza prawdopodobieństwo tego, że  $h(x_1) = h(x_2)$ , gdzie  $x_1$  i  $x_2$  są losowymi (niekoniecznie różnymi) elementami  $X$ . Udogodnij, że

$$\epsilon \geq \frac{1}{|Y|},$$

przy czym równość zachodzi wtedy i tylko wtedy, gdy

$$|h^{-1}(y)| = \frac{|X|}{|Y|}$$

dla każdego  $y \in Y$ .

- 7.3. Rozważmy funkcję skrótu Chauma–van Heijsta–Pfitzmannego z wartościami  $p = 15083$ ,  $\alpha = 154$  i  $\beta = 2307$ . Mając konflikt

$$\alpha^{7431}\beta^{5564} \equiv \alpha^{1459}\beta^{954} \pmod{p},$$

oblicz  $\log_{\alpha}\beta$ .

- 7.4. Niech  $n = pq$ , gdzie  $p$  i  $q$  są dwiema różnymi (i tajnymi) dużymi liczbami pierwszymi, takimi że  $p = 2p_1 + 1$  oraz  $q = 2q_1 + 1$  dla pewnych liczb pierwszych  $p_1$  i  $q_1$ . Niech ponadto  $\alpha$  będzie elementem rzędu  $2p_1q_1$  w  $\mathbb{Z}_n^*$  (jest to największy możliwy rzząd elementu w  $\mathbb{Z}_n^*$ ). Określmy funkcję skrótu  $h : \{1, \dots, n^2\} \rightarrow \mathbb{Z}_n^*$  za pomocą wzoru  $h(x) = \alpha^x \pmod{n}$ .

Przyjmijmy, że do zdefiniowania takiej funkcji  $h$  wybrano wartości  $n = 603241$  oraz  $\alpha = 11$  i założymy, że mamy trzy konflikty funkcji  $h$ :  $h(1294755) = h(80115359) = h(52738737)$ . Wykorzystaj te informacje do rozłożenia na czynniki pierwsze liczby  $n$ .

- 7.5. Niech  $h_1 : (\mathbb{Z}_2)^{2m} \rightarrow (\mathbb{Z}_2)^m$  będzie silnie bezkonfliktową funkcją skrótu.

- (a) Określ funkcję  $h_2 : (\mathbb{Z}_2)^{4m} \rightarrow (\mathbb{Z}_2)^m$  tak jak na rysunku 7.13. Wykaż, że  $h_2$  jest silnie bezkonfliktowa.

- (1) zapisz  $x \in (\mathbb{Z}_2)^{4m}$  jako  $x = x_1 \parallel x_2$ , gdzie  $x_1, x_2 \in (\mathbb{Z}_2)^{2m}$   
 (2) zdefiniuj  $h_2(x) = h_1(h_1(x_1) \parallel h_1(x_2))$

RYSUNEK 7.13. Skracanie  $4m$  bitów do  $m$  bitów

- (b) Dla pewnej liczby całkowitej  $i$  określmy rekurencyjnie funkcję skrótu  $h_i : (\mathbb{Z}_2)^{2^im} \rightarrow (\mathbb{Z}_2)^m$ , tak jak to widać na rysunku 7.14. Udowodnij, że  $h_i$  jest silnie bezkonfliktowa.

- (1) zapisz  $x \in (\mathbb{Z}_2)^{2^im}$  jako  $x = x_1 \parallel x_2$ , gdzie  $x_1, x_2 \in (\mathbb{Z}_2)^{2^{i-1}m}$   
 (2) zdefiniuj  $h_i(x) = h_1(h_{i-1}(x_1) \parallel h_{i-1}(x_2))$

RYSUNEK 7.14. Skracanie  $2^im$  bitów do  $m$  bitów

- 7.6. Użyj (oryginalnej) funkcji rozszerzania SHS opisanej równaniem (7.1) do wyrażenia ciągów  $X[16], \dots, X[79]$  za pomocą  $X[0], \dots, X[15]$ . Dla każdej pary  $X[i], X[j]$ , gdzie  $1 \leq i < j \leq 15$ , użyj programu komputerowego do obliczenia  $\lambda_{ij}$ , czyli liczby tych ciągów  $X[k]$  ( $16 \leq k \leq 79$ ), w których reprezentacji występują zarówno  $X[i]$ , jak i  $X[j]$ . Jaki jest zakres wartości  $\lambda_{ij}$ ?

# 8

---

## Uzgadnianie i dystrybucja klucza

---

### 8.1. Wprowadzenie

Stwierdziliśmy, że systemy z kluczem publicznym mają tę przewagę nad systemami z kluczem prywatnym, że nie wymagają bezpiecznego kanału do przekazania klucza tajnego. Niestety, większość tych pierwszych systemów działa znacznie wolniej niż te drugie, jak na przykład DES. Dlatego w praktyce systemy z kluczem prywatnym są zazwyczaj stosowane do szyfrowania „długich” wiadomości. W ten sposób powraca jednak problem przesyłania kluczy tajnych.

W tym rozdziale omówimy różne podejścia do zagadnienia ustalania klucza tajnego. Wprowadzimy rozróżnienie między dystrybucją klucza a jego uzgadnieniem. *Dystrybucją klucza* nazwiemy proces, w ramach którego jedna ze stron wybiera klucz tajny i przekazuje go drugiej lub drugim stronom, natomiast *uzgadnianie klucza* oznacza protokół ustalania przez dwie lub więcej stron klucza tajnego, przy czym strony komunikują się za pośrednictwem publicznego kanału. W schemacie uzgadniania klucza jego wartość określa się na podstawie danych dostarczonych przez obie strony.

Kontekstem naszych rozważań będzie niezapewniająca bezpieczeństwa sieć  $n$  użytkowników. W niektórych przypadkach będziemy zakładać istnienie *warygodnego czynnika* (oznaczanego skrótem TA od ang. *trusted authority*), odpowiedzialnego za ustalanie tożsamości użytkowników, wybór i przekazanie klucza użytkownikom itp.

Ponieważ sieć nie jest wystarczająco bezpieczna, musimy zapewnić jej ochronę przed potencjalnymi wrogami. Nasz przeciwnik Oskar może się okazać przeciwnikiem *biernym*, co oznacza, że jego działania ograniczają się do podsłuchiwania wiadomości przesyłanych kanałem. Jednak chcielibyśmy się również zabezpieczyć przed możliwością, że Oskar będzie przeciwnikiem *aktywnym*. Aktywny przeciwnik może podejmować wiele przykrych akcji. Może na przykład czynić starania by:

- (1) zmodyfikować wiadomości przechwycone w sieci;

- (2) zapisać wiadomości w celu późniejszego wykorzystania;
- (3) podszyć się pod różnych użytkowników sieci.

Przeciwnik aktywny może próbować osiągnąć jeden z dwóch celów:

- (1) wprowadzenie w błąd użytkowników U i V, tak aby uznali „nieważny” klucz za poprawny (nieważny może być stary klucz, który utracił swoją ważność, albo klucz wybrany przez przeciwnika, żeby wymienić tylko dwie możliwości).
- (2) spowodowanie, by U lub V uwierzyli, że dokonali wymiany klucza, choć w rzeczywistości do tego nie doszło.

Proces dystrybucji klucza oraz protokoł jego uzgadniania mają doprowadzić do tego, aby po realizacji protokołu obie zaangażowane strony dysponowały tym samym kluczem  $K$ , którego wartość nie jest znana nikomu innemu (oprócz, być może, TA). Rzecz jasna, zaprojektowanie protokołu zapewniającego tego rodzaju bezpieczeństwo jest znacznie trudniejsze, gdy mamy do czynienia z przeciwnikiem aktywnym, niż wtedy, gdy jest to przeciwnik bierny.

Najpierw w podrozdziale 8.2 rozważymy pojęcie *wstępnej dystrybucji* klucza. Dla każdej pary użytkowników TA wybiera losowo klucz  $K_{U,V} = K_{V,U}$  i przekazuje go poza siecią użytkownikom U i V, wykorzystując do tego bezpieczny kanał (poza siecią, gdyż sieć nie zapewnia bezpieczeństwa). To podejście daje bezwarunkowe bezpieczeństwo, wymaga jednak bezpiecznego kanału do przesyłania informacji między TA i każdym użytkownikiem. Był może jednak większe znaczenie ma fakt, że użytkownik musi przechować  $n - 1$  kluczy, podczas gdy TA musi bezpiecznie przekazać ogółem  $\binom{n}{2}$  kluczy (niekiedy nazywa się to „problemem  $n^2$ ”). Nawet dla stosunkowo małych sieci koszt tej procedury jest niewspółmiernie duży, nie jest to więc rozwiązanie, które można stosować w praktyce.

W punkcie 8.2.1 omawiamy interesujący, bezwarunkowo bezpieczny schemat wstępnej dystrybucji klucza. Zaproponowany przez Bloma schemat pozwala na zredukowanie ilości tajnej informacji, którą muszą przechowywać użytkownicy sieci. W punkcie 8.2.2 przedstawimy pewien obliczeniowo bezpieczny schemat wstępnej dystrybucji klucza, który jest oparty na problemie logarytmu dyskretnego.

Bardziej praktyczne podejście można by określić jako *dystrybucja klucza przez TA on-line*. W takim schemacie TA pełni rolę *serwera kluczy*. Z każdym użytkownikiem sieci U dzieli on wspólny tajny klucz  $K_U$ . Gdy U chce się komunikować z V, zwraca się do TA o *klucz sesji*. TA generuje klucz sesji  $K$  i przekazuje go w postaci zaszyfrowanej do U i V, a ci go u siebie deszyfrują. Na takim podejściu jest oparty znany system Kerberos, którym zajmiemy się w podrozdziale 8.3.

Jeśli z jakichś względów współpraca on-line z TA jest niepraktyczna lub niepożądana, można odwołać się do często stosowanej metody *protokołu uzgadniania klucza*. Przy takim podejściu użytkownicy U i V wspólnie wybierają klucz, komunikując się za pośrednictwem publicznego kanału. Ten godny uwagi pomysł pochodzi od Diffiego i Hellmana, a także (niezależnie) od Merkle'a. Opisze-

my tu kilka najczęściej stosowanych protokołów uzgadniania klucza. W punkcie 8.4.1 przedstawimy wariant oryginalnego protokołu Diffiego i Hellmana ze wzmoczoną ochroną przed aktywnym przeciwnikiem. Omówimy również dwa inne ciekawe protokoły: schemat MTI w punkcie 8.4.2 oraz schemat Giraulta w punkcie 8.4.3.

---

## 8.2. Wstępna dystrybucja klucza

W podstawowym wariantie metody TA generuje  $\binom{n}{2}$  kluczy, przekazując każdy z nich jednej parze użytkowników sieci (dzielonej przez  $n$  użytkowników). Jak stwierdziliśmy wcześniej, do przekazania kluczy potrzebujemy bezpiecznego kanału między TA a każdym z użytkowników. To istotny postęp w stosunku do sytuacji, w której każda para użytkowników niezależnie wymienia między sobą klucze za pośrednictwem bezpiecznego kanału, ponieważ dzięki temu można ograniczyć liczbę niezbędnych bezpiecznych połączeń z  $\binom{n}{2}$  do  $n$ . Jeśli jednak liczba  $n$  jest duża, to rozwiązanie też nie jest dość praktyczne, zarówno z punktu widzenia ilości informacji, którą należy bezpiecznie przekazać, jak i z punktu widzenia informacji, którą każdy użytkownik musi bezpiecznie przechować (a więc liczby tajnych kluczy pozostałych  $n - 1$  użytkowników).

Warto zatem podjąć próbę zredukowania ilości przekazywanej i przechowywanej informacji, nie rezygnując z tego, by każda para użytkowników U i V mogła obliczyć (niezależnie) swój klucz tajny  $K_{U,V}$ . W kolejnym punkcie omówimy elegancki schemat, zwany schematem wstępnej dystrybucji klucza Bloma, który pozwala to osiągnąć.

### 8.2.1. Schemat Bloma

Jak poprzednio nasze rozważania dotyczą sieci z  $n$  użytkownikami. Dla wygody przyjmiemy, że klucze są wybierane ze skończonego ciała  $\mathbb{Z}_p$ , gdzie  $p \geq n$  jest liczbą pierwszą. Niech  $k$  będzie liczbą całkowitą,  $1 \leq k \leq n - 2$ . Wartość  $k$  określa maksymalną wielkość koalicji, której atakowi system potrafi się oprzeć. W **schemacie Bloma** TA przekazuje bezpiecznym kanałem  $k + 1$  elementów  $\mathbb{Z}_p$  każdemu użytkownikowi (wobec  $n - 1$  elementów przekazywanych w wariantie podstawowym). Tu także każda para użytkowników U i V może obliczyć klucz  $K_{U,V} = K_{V,U}$ . Warunek bezpieczeństwa jest teraz następujący: żaden zbiór co najwyżej  $k$  użytkowników, rozłączny z  $\{U, V\}$ , nie jest w stanie ustalić jakiejkolwiek informacji o kluczu  $K_{U,V}$  (mówimy tu o bezpieczeństwie bezwarunkowym).

Najpierw przedstawimy szczególny przypadek schematu Bloma, gdy  $k = 1$ . Teraz TA przekazuje każdemu użytkownikowi dwa elementy ze zbioru  $\mathbb{Z}_p$  bezpiecznym kanałem, a żaden indywidualny użytkownik sieci W nie jest w stanie ustalić jakiejkolwiek informacji o  $K_{U,V}$ , o ile  $W \neq U, V$ . Schemat Bloma jest przedstawiony na rysunku 8.1, a następujący przykład ilustruje jego działanie.

- (1) podaje się do publicznej wiadomości liczbę pierwszą  $p$ , a także przydzielony każdemu użytkownikowi element  $r_U \in \mathbb{Z}_p$ ; elementy  $r_U$  muszą być różne.
- (2) TA wybiera losowe elementy  $a, b, c \in \mathbb{Z}_p$  (niekoniecznie różne) i tworzy wielomian

$$f(x, y) = a + b(x + y) + cxy \bmod p$$

- (3) dla każdego użytkownika U TA oblicza wielomian

$$g_U(x) = f(x, r_U) \bmod p$$

i bezpiecznym kanałem przekazuje mu  $g_U(x)$ ; zauważmy, że  $g_U(x)$  jest wielomianem liniowym ze względu na  $x$ , można go więc zapisać w postaci

$$g_U(x) = a_U + b_U x,$$

gdzie

$$a_U = a + br_U \bmod p$$

oraz

$$b_U = b + cr_U \bmod p$$

- (4) gdy U i V zechcą się skomunikować, użyją wspólnego klucza

$$K_{U,V} = K_{V,U} = f(r_U, r_V) = a + b(r_U + r_V) + cr_U r_V \bmod p,$$

przy czym U oblicza  $K_{U,V}$  jako

$$f(r_U, r_V) = g_U(r_V),$$

podczas gdy V oblicza  $K_{U,V}$  jako

$$f(r_U, r_V) = g_V(r_U)$$

**RYSUNEK 8.1.** Schemat dystrybucji klucza Bloma ( $k = 1$ )

### Przykład 8.1

Załóżmy, że mamy trzech użytkowników: U, V i W,  $p = 17$ , a publicznymi elementami użytkowników są:  $r_U = 12$ ,  $r_V = 7$  oraz  $r_W = 1$ . Przypuśćmy, że TA wybiera  $a = 8$ ,  $b = 7$  i  $c = 2$ , a więc wielomian  $f$  ma postać

$$f(x, y) = 8 + 7(x + y) + 2xy.$$

Wielomianami  $g$  są:

$$g_U(x) = 7 + 14x$$

$$g_V(x) = 6 + 4x$$

$$g_W(x) = 15 + 9x.$$

W rezultacie mamy trzy klucze:

$$K_{U,V} = 3$$

$$K_{U,W} = 4$$

$$K_{V,W} = 10.$$

Użytkownik U obliczy  $K_{U,V}$  w sposób następujący:

$$g_U(r_V) = 7 + 14 \cdot 7 \bmod 17 = 3,$$

Użytkownik V obliczy  $K_{U,V}$  jako

$$g_V(r_U) = 6 + 4 \cdot 12 \bmod 17 = 3.$$

Obliczenie pozostałych kluczy pozostawiamy Czytelnikowi jako ćwiczenie.  $\square$

Udowodnimy teraz, że żaden użytkownik nie potrafi samodzielnie uzyskać jakiekolwiek informacji o kluczach pozostałych dwóch użytkowników.

### **TWIERDZENIE 8.1**

Schemat Bloma dla  $k = 1$  jest bezwarunkowo bezpieczny wobec dowolnego pojedynczego użytkownika.

**DOWÓD** Przypuśćmy, że W chce obliczyć klucz

$$K_{U,V} = a + b(r_U + r_V) + cr_Ur_V \bmod p.$$

Wartości  $r_U$  i  $r_V$  są znane publicznie, nieznane są natomiast liczby  $a$ ,  $b$  i  $c$ . Użytkownik W zna wartości

$$a_W = a + br_W \bmod p$$

oraz

$$b_W = b + cr_W \bmod p,$$

ponieważ są to współczynniki wielomianu  $g_W(x)$  przysłanego użytkownikowi W przez TA.

Wykażemy teraz, że informacja znana użytkownikowi W jest zgodna z każdą możliwą wartością  $\ell \in \mathbb{Z}_p$  klucza  $K_{U,V}$ , co oznacza, że użytkownik W żadnej z nich nie może odrzucić. Rozważmy następujące równanie macierzowe (w  $\mathbb{Z}_p$ ):

$$\begin{pmatrix} 1 & r_U + r_V & r_Ur_V \\ 1 & r_W & 0 \\ 0 & 1 & r_W \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \ell \\ a_W \\ b_W \end{pmatrix}.$$

Pierwsze równanie reprezentuje hipotezę, że  $K_{U,V} = \ell$ ; drugie i trzecie zawierają informację o  $a$ ,  $b$  i  $c$ , którą W zna na podstawie znajomości wielomianu  $g_W(x)$ .

Wyznacznik macierzy współczynników ma postać

$$r_W^2 + r_U r_V - (r_U + r_V) r_W = (r_W - r_U)(r_W - r_V),$$

przy czym wszystkie operacje arytmetyczne są prowadzone w  $\mathbb{Z}_p$ . Ponieważ  $r_W \neq r_U$  i  $r_W \neq r_V$ , macierz współczynników układu równań ma niezerowy wyznacznik, a więc istnieje jednoznaczne rozwiązanie ze względu na  $a$ ,  $b$  i  $c$ . Innymi słowy, każda możliwa wartość  $\ell$  klucza  $K_{U,V}$  jest zgodna z informacją posiadaną przez W. ■

Jednakże koalicja dwóch użytkowników, powiedzmy {W,X}, może ustalić wartość dowolnego klucza  $K_{U,V}$ , gdzie  $\{W,X\} \cap \{U,V\} = \emptyset$ . Razem W i X wiedzą, że

$$a_W = a + b r_W$$

$$b_W = b + c r_W$$

$$a_X = a + b r_X$$

$$b_X = b + c r_X.$$

Mają zatem cztery równania z trzema niewiadomymi, mogą więc łatwo obliczyć jednoznaczne rozwiązanie dla  $a$ ,  $b$  i  $c$ . Znając te wartości, potrafią zbudować wielomian  $f(x,y)$  i obliczyć każdy klucz, który zechcą.

Taki schemat łatwo uogólnić, tak by pozostał bezpieczny wobec koalicji  $k$  użytkowników; zmienia się tylko krok 2. TA używa wielomianu  $f(x,y)$  postaci

$$f(x,y) = \sum_{i=0}^k \sum_{j=0}^k a_{i,j} x^i y^j \bmod p,$$

gdzie  $a_{i,j} \in \mathbb{Z}_p$  ( $0 \leq i \leq k$ ,  $0 \leq j \leq k$ ) oraz  $a_{i,j} = a_{j,i}$  dla wszystkich  $i$ ,  $j$ . Pozostałe składniki protokołu pozostają bez zmian.

### 8.2.2. Schemat wstępnej dystrybucji klucza Diffiego–Hellmana

Opiszemy tu schemat stanowiący modyfikację powszechnie znanego protokołu wymiany klucza Diffiego–Hellmana, który omówimy nieco później, w podrozdziale 8.4. Nazywa się go schematem wstępnej dystrybucji klucza Diffiego–Hellmana. Jest on obliczeniowo bezpieczny, jeśli tylko pewien problem związany z logarytmem dyskretnym jest obliczeniowo nieroziwiążalny.

Będziemy rozważać schemat nad  $\mathbb{Z}_p$ , gdzie  $p$  jest liczbą pierwszą, choć można go realizować w dowolnej grupie skończonej, w której problem logarytmu dyskretnego jest obliczeniowo nieroziwiążalny. Przyjmiemy, że  $\alpha$  jest elementem pierwotnym grupy  $\mathbb{Z}_p$  oraz że wartości  $p$  i  $\alpha$  są publicznie znane każdemu użytkownikowi sieci.

Niech dla każdego użytkownika  $U$  w tym schemacie  $ID(U)$  oznacza identyfikującą go informację, na przykład nazwisko, adres elektroniczny, numer telefonu lub inne istotne dane. Ponadto, każdy użytkownik  $U$  ma tajny wykładnik  $a_U$  (gdzie  $0 \leq a_U \leq p - 2$ ) oraz odpowiednią wartość publiczną

$$b_U = \alpha^{a_U} \bmod p.$$

Z kolei TA dysponuje schematem podpisu  $z$  (publicznym) algorytmem weryfikacji  $ver_{TA}$  i tajnym algorytmem podpisu  $sig_{TA}$ . Przyjmiemy także milczące założenie, że każda informacja jest przed podpisaniem poddana skracaniu za pomocą publicznej funkcji skrótu. Dla uproszczenia opisu procedury przy omawianiu protokołu będziemy pomijać proces skracania.

Część informacji użytkownika  $U$  będzie uwiarygodniana *certyfikatem* wydanym i podpisywanym przez TA. Każdy użytkownik  $U$  będzie posiadał certyfikat

$$C(U) = (ID(U), b_U, sig_{TA}(ID(U), b_U)),$$

gdzie  $b_U$  powstaje w sposób przedstawiony wyżej (zauważmy, że TA nie musi znać wartości  $a_U$ ). Użytkownik  $U$  dostaje certyfikat, gdy dołącza się do sieci. Certyfikaty mogą być przechowywane w publicznej bazie danych albo każdy użytkownik przechowuje swój certyfikat u siebie. Podpis TA na certyfikacie pozwala każdemu użytkownikowi sieci na sprawdzenie informacji, którą ten certyfikat zawiera.

Dwaj użytkownicy  $U$  i  $V$  mogą bardzo łatwo obliczyć wspólny klucz

$$K_{U,V} = \alpha^{a_U a_V} \bmod p,$$

jak widać na rysunku 8.2.

- (1) do wiadomości publicznej podaje się liczbę pierwszą  $p$  oraz element pierwotny  $\alpha \in \mathbb{Z}_p^*$
  - (2)  $V$  oblicza

$$K_{U,V} = \alpha^{a_U a_V} \bmod p = b_U^{a_V} \bmod p$$

używając publicznej wartości  $b_U$ , pochodzącej z certyfikatu  $U$ , wraz z własną tajną wartością  $a_V$

- (3)  $U$  oblicza

$$K_{U,V} = \alpha^{a_U a_V} \bmod p = b_V^{a_U} \bmod p,$$

używając publicznej wartości  $b_V$ , pochodzącej z certyfikatu  $V$ , wraz ze swoją własną tajną wartością  $a_U$

**RYSUNEK 8.2. Schemat wstępnej dystrybucji klucza Diffiego-Hellmana**

Ilustracją algorytmu niech będzie następujący przykład.

### Przykład 8.2

Załóżmy, że wartości  $p = 25307$  i  $\alpha = 2$  są publicznie znane ( $p$  jest liczbą pierwszą,  $\alpha$  zaś pierwiastkiem pierwotnym modulo  $p$ ). Przypuśćmy, że U wybiera  $a_U = 3578$ . Wówczas oblicza

$$\begin{aligned} b_U &= \alpha^{a_U} \bmod p \\ &= 2^{3578} \bmod 25307 \\ &= 6113, \end{aligned}$$

i umieszcza tę wartość w swoim certyfikacie. Jeśli wartością wybraną przez V jest  $a_V = 19956$ , to obliczenia prowadzą do

$$\begin{aligned} b_V &= \alpha^{a_V} \bmod p \\ &= 2^{19956} \bmod 25307 \\ &= 7984, \end{aligned}$$

i ta wartość trafia do certyfikatu V.

Teraz U może obliczyć klucz

$$\begin{aligned} K_{U,V} &= b_V^{a_U} \bmod p \\ &= 7984^{3578} \bmod 25307 \\ &= 3694. \end{aligned}$$

Również V może obliczyć ten sam klucz:

$$\begin{aligned} K_{U,V} &= b_U^{a_V} \bmod p \\ &= 6113^{19956} \bmod 25307 \\ &= 3694. \end{aligned}$$

□

Zajmijmy się teraz bezpieczeństwem systemu przy założeniu obecności biernego lub aktywnego przeciwnika. Podpis TA na certyfikacie użytkownika skutecznie zapobiega ewentualnym zmianom w informacji, które W mógłby poczytać w cudzym certyfikacie. Wystarczy zatem, jeśli ograniczymy się do rozważenia ataków biernych. Chodzi tu więc o odpowiedź na pytanie: czy użytkownik W może obliczyć  $K_{U,V}$ , gdy  $W \neq U, V$ ? Inaczej mówiąc, czy mając dane  $\alpha^{a_U} \bmod p$  oraz  $\alpha^{a_V} \bmod p$  (ale nie  $a_U$  ani  $a_V$ ), można obliczyć  $\alpha^{a_U a_V} \bmod p$ ? Ten problem nosi nazwę **problemu Diffiego–Hellmana**. Definiujemy go formalnie (w postaci równoważnej, choć nieco odmiennej) na rysunku 8.3. Oczywiście schemat wstępnej dystrybucji klucza Diffiego–Hellmana jest bezpieczny w przypadku ataku biernego przeciwnika wtedy i tylko wtedy, gdy problem Diffiego–Hellmana jest obliczeniowo nieroziągalny.

Gdyby W potrafił obliczyć  $a_U$  na podstawie znajomości  $b_U$  albo gdyby umiał ustalić wartość  $a_V$ , dysponując  $b_V$ , mógłby obliczyć  $K_{U,V}$  dokładnie tak, jak obliczyłby ten klucz U (lub V). Jednakże oba te obliczenia są szczególnymi przypadkami problemu logarytmu dyskretnego. Tak więc, jeśli problem logarytmu

**Dane**  $I = (p, \alpha, \beta, \gamma)$ , gdzie  $p$  jest liczbą pierwszą,  $\alpha \in \mathbb{Z}_p^*$  jest elementem pierwotnym oraz  $\beta, \gamma \in \mathbb{Z}_p^*$ .

**Cel** Obliczyć  $\beta^{\log_\alpha \gamma} \bmod p$  ( $= \gamma^{\log_{\alpha \beta} \beta} \bmod p$ ).

### RYSUNEK 8.3. Problem Diffiego–Hellmana

dyskretnego jest obliczeniowo nieroziągalny w  $\mathbb{Z}_p$ , to schemat wstępnej dystrybucji klucza Diffiego–Hellmana jest bezpieczny wobec tego rodzaju ataków. Istnieje jednak nieudowodnione przypuszczenie, że dowolny algorytm rozwiązuający problem Diffiego–Hellmana mógłby posłużyć także do rozwiązywania problemu logarytmu dyskretnego. (Przypomina to sytuację z RSA, gdyż przypuszcza się, choć nie ma na to dowodu, że złamanie RSA jest wielomianowo równoważne faktoryzacji).

Jak wynika z powyższych uwag, problem Diffiego–Hellmana nie jest trudniejszy od problemu logarytmu dyskretnego. Jeśli nawet nie potrafimy określić dokładnie poziomu jego trudności, możemy wskazać związek między jego bezpieczeństwem a bezpieczeństwem innego, znanego nam już systemu kryptograficznego, a mianowicie systemu ElGamala.

### TWIERDZENIE 8.2

Złamanie systemu kryptograficznego ElGamala jest równoważne rozwiązywaniu problemu Diffiego–Hellmana.

**DOWÓD** Przypomnijmy najpierw, jak działa w systemie ElGamala szyfrowanie i deszyfrowanie. Kluczem jest  $K = (p, \alpha, a, \beta)$ , gdzie  $\beta = \alpha^a \bmod p$  ( $a$  jest tajne, a  $p$ ,  $\alpha$  i  $\beta$  – publiczne). Dla (tajnej) liczby losowej  $k \in \mathbb{Z}_{p-1}$  mamy

$$e_K(x, k) = (y_1, y_2),$$

gdzie

$$y_1 = \alpha^k \bmod p$$

oraz

$$y_2 = x\beta^k \bmod p.$$

Dla  $y_1, y_2 \in \mathbb{Z}_p^*$  zachodzi

$$d_K(y_1, y_2) = y_2(y_1^a)^{-1} \bmod p.$$

Przypuśćmy, że **A** jest algorytmem rozwiązującym problem Diffiego–Hellmana i mamy dany tekst zaszyfrowany  $(y_1, y_2)$  w systemie ElGamala. Zastosujemy algorytm **A** z danymi wejściowymi  $p, \alpha, y_1, \beta$ . Otrzymujemy wartość

$$\begin{aligned} \mathbf{A}(p, \alpha, y_1, \beta) &= \mathbf{A}(p, \alpha, \alpha^k, \alpha^a) \\ &= \alpha^{ka} \bmod p \\ &= \beta^k \bmod p. \end{aligned}$$

Teraz można już łatwo deszyfrować  $(y_1, y_2)$ :

$$x = y_2(\beta^k)^{-1} \bmod p.$$

I na odwrót, przypuśćmy, że mamy algorytm **B**, który realizuje deszyfrowanie w systemie ElGamala. Tak więc **B** pobiera dane wejściowe  $p, \alpha, \beta, y_1$  oraz  $y_2$  i oblicza

$$x = y_2(y_1^{\log_\alpha \beta})^{-1} \bmod p.$$

Łatwo zauważyc, że biorąc dane wejściowe  $p, \alpha, \beta$  i  $\gamma$ , dla problemu Diffiego-Hellmana mamy

$$\begin{aligned} \mathbf{B}(p, \alpha, \beta, \gamma, 1)^{-1} &= 1((\gamma^{\log_\alpha \beta})^{-1})^{-1} \bmod p \\ &= \gamma^{\log_\alpha \beta} \bmod p, \end{aligned}$$

a więc to, co chcieliśmy wykazać. ■

### 8.3. Kerberos

W metodach wstępnej dystrybucji klucza, o których była dotąd mowa, każda para użytkowników oblicza dla siebie jeden ustalony klucz. Jeżeli używa się tego samego klucza przez dłuższy czas, rośnie niebezpieczeństwo jego kompromitacji. Dlatego częściej stosować zasadę, że każde sieciowe połączenie między dwoma użytkownikami wymaga odrębnego klucza sesji (tę zasadę nazywa się *świeżością klucza*; ang. *key freshness*).

Gdy przekazuje się klucze on-line, nie ma potrzeby, by każdy użytkownik przechowywał klucze do komunikowania się z innymi użytkownikami (choć każdy będzie dzielił klucz z TA). Klucze sesji mogą być na żądanie przekazywane przez TA i na TA spoczywa odpowiedzialność za odświeżanie kluczy.

**Kerberos** jest popularnym systemem dostarczania klucza (ang. *key serving system*) opartym na kriptografii z kluczem prywatnym. W tym podrozdziale zajmiemy się protokołem wydawania kluczy sesji w systemie Kerberos. Każdy użytkownik U dzieli z TA tajny klucz DES  $K_U$ . W najnowszej wersji systemu Kerberos (wersja V) wszystkie przekazywane wiadomości są szyfrowane w trybie CBC, który opisaliśmy w punkcie 3.4.1.

Podobnie jak w punkcie 8.2.2, ID(U) oznacza tu publiczną informację identyfikującą użytkownika U. Gdy do TA dociera żądanie wydania klucza sesji, ten generuje nowy, losowy klucz  $K$ . Ponadto, TA rejestruje w postaci *datownika* czas  $T$ , w którym takie żądanie wpłynęło, oraz określa *okres ważności L* (ang. *lifetime*), wskazujący jak długo klucz  $K$  może być stosowany. Tak więc klucz sesji  $K$  jest aktualny wyłącznie od momentu  $T$  do  $T + L$ . Cała ta informacja jest szyfrowana i przekazywana do użytkownika U, a w końcu także do V. Zanim przejdziemy do szczegółów, przedstawiamy protokół na rysunku 8.4.

- (1) U prosi TA o klucz sesji do komunikacji z V  
 (2) TA wybiera losowy klucz sesji  $K$ , datownik  $T$  oraz okres ważności  $L$   
 (3) TA oblicza

$$m_1 = e_{K_U}(K, \text{ID(V)}, T, L)$$

oraz

$$m_2 = e_{K_V}(K, \text{ID(U)}, T, L),$$

po czym przekazuje  $m_1$  i  $m_2$  użytkownikowi U

- (4) korzystając z funkcji  $d_{K_U}$ , U oblicza  $K$ ,  $T$ ,  $L$  oraz  $\text{ID(V)}$  na podstawie wartości  $m_1$ ; dalej liczy

$$m_3 = e_K(\text{ID(U)}, T)$$

i przekazuje  $m_3$  użytkownikowi V wraz z wiadomością  $m_2$  otrzymaną od TA

- (5) korzystając z funkcji  $d_{K_V}$ , V oblicza  $K$ ,  $T$ ,  $L$  oraz  $\text{ID(U)}$  na podstawie wartości  $m_2$ ; następnie, znając  $m_3$ , za pomocą funkcji  $d_K$  oblicza  $T$  i  $\text{ID(U)}$ ; sprawdza, czy obie otrzymane wartości  $T$  oraz obie wartości  $\text{ID(U)}$  są równe i jeśli tak, oblicza

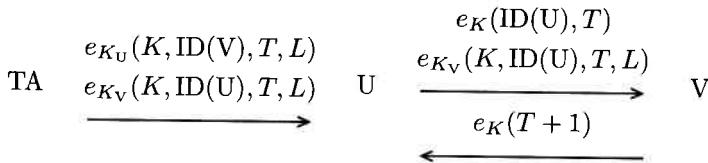
$$m_4 = e_K(T + 1)$$

po czym przekazuje wynik do U

- (6) U deszyfruje  $m_4$  za pomocą funkcji  $d_K$  i sprawdza, czy wynik jest równy  $T + 1$

#### RYSUNEK 8.4. Przekazanie klucza sesji w systemie Kerberos

Poniższy diagram ilustruje przepływ informacji w ramach protokołu:



Wyjaśnijmy teraz, co się dzieje w poszczególnych krokach realizacji protokołu. Nie mamy co prawda formalnego dowodu bezpieczeństwa systemu Kerberos w przypadku działań aktywnego przeciwnika, możemy jednak przynajmniej przytoczyć nieformalne argumenty dotyczące cech protokołu.

Jak wspomnieliśmy wcześniej, TA generuje  $K$ ,  $T$  i  $L$  w kroku 2. W kroku 3 ta informacja, razem z  $\text{ID(V)}$ , jest szyfrowana za pomocą wspólnego dla U i TA klucza  $K_U$  do postaci  $m_1$ . Podobnie  $K$ ,  $T$ ,  $L$  i  $\text{ID(U)}$  są szyfrowane za pomocą

wspólnego dla V i TA klucza  $K_V$  do postaci  $m_2$ . Obie te zaszyfrowane wiadomości są przesyłane do U.

Użytkownik U może użyć swojego klucza do odczytania  $m_1$  i otrzyma w ten sposób wartości  $K$ ,  $T$  i  $L$ . Sprawdza wtedy, czy aktualny czas mieści się w przedziale między  $T$  i  $T + L$ . Może także sprawdzić, czy klucz sesji  $K$  został wysłany do żądanego partnera V, weryfikując informację  $ID(V)$  odczytaną z  $m_1$ .

Następnie U przesyła  $m_2$  do V, a jednocześnie za pomocą nowego klucza sesji  $K$  szyfruje  $T$  oraz  $ID(U)$  i powstającą w ten sposób wiadomość  $m_3$  również wysyła do V.

Gdy V otrzymuje  $m_2$  i  $m_3$  od użytkownika U, deszyfruje  $m_2$ , odczytuje  $T$ ,  $K$ ,  $L$  oraz  $ID(U)$ , po czym za pomocą nowego klucza sesji  $K$  deszyfruje  $m_3$  i sprawdza, czy wartości  $T$  i  $ID(U)$ , odczytane niezależnie z  $m_2$  i  $m_3$ , pokrywają się. To go upewnia, że klucz sesji odczytany z wiadomości  $m_2$  jest tym samym kluczem, którego użyto do szyfrowania  $m_3$ . Dalej V szyfruje  $T + 1$  za pomocą klucza  $K$  i przekazuje wynik użytkownikowi U w postaci wiadomości  $m_4$ .

Użytkownik U otrzymuje wiadomość  $m_4$  i odczytując ją za pomocą klucza  $K$ , sprawdza, czy jej treścią jest  $T + 1$ . To go upewnia, że klucz sesji  $K$  dotarł bezpiecznie do V, ponieważ klucz ten jest niezbędny do utworzenia tej wiadomości.

Warto podkreślić tu różnorodność funkcji, które spełniają wiadomości przesyłane w ramach protokołu. Wiadomości  $m_1$  i  $m_2$  służą do zapewnienia tajności przy przekazywaniu klucza sesji  $K$ . Z kolei  $m_3$  i  $m_4$  umożliwiają *potwierdzenie klucza* (ang. *key confirmation*), czyli pozwalają U i V przekonać się nawzajem o posiadaniu tego samego klucza sesji  $K$ . W większości schematów dystrybucji klucza potwierdzenie klucza (sesji) można zawsze dołączyć, nawet jeśli nie występuje w oryginalnym schemacie. Zwykle odbywa się to podobnie jak w systemie Kerberos, mianowicie przez użycie nowego klucza sesji  $K$  do zaszyfrowania znanych wielkości. W Kerberos użytkownik U stosuje klucz  $K$  do zaszyfrowania  $ID(U)$  i  $T$ , które już wcześniej zostały zaszyfrowane w wiadomości  $m_2$ . Podobnie V używa klucza  $K$  do zaszyfrowania  $T + 1$ . Datownik  $T$  i okres ważności  $L$  stosuje się po to, by zapobiec przechowywaniu „starych” wiadomości przez aktywnego przeciwnika w celu ich późniejszego ponownego wykorzystania (co określa się mianem *ataku z powtórzeniem* (ang. *replay attack*). Skuteczność tej metody obrony wynika z zasady odrzucania kluczy, których ważność wygasła.

Jedną z wad systemu Kerberos jest konieczność synchronizacji zegarków wszystkich użytkowników sieci, ponieważ o terminie ważności klucza sesji decyduje bieżący czas. W praktyce bardzo trudno zapewnić pełną synchronizację, trzeba zatem dopuścić pewien zakres rozbieżności.

---

## 8.4. Protokół wymiany kluczy Diffiego–Hellmana

Jeśli nie chcemy korzystać z serwera kluczy działającego on-line, musimy do wymiany kluczy tajnych użyć protokołu uzgadniania klucza. Pierwszym, a zarazem najlepiej znanym takim protokołem jest **protokół wymiany klucza Diffiego–**

**-Hellmana.** Zakłada się w nim, że  $p$  jest liczbą pierwszą,  $\alpha$  jest elementem pierwotnym w  $\mathbb{Z}_p$ , przy czym wartości  $p$  i  $\alpha$  są publicznie znane. (Alternatywnie, wartości te mogłyby być wybrane przez użytkownika U i przekazane użytkownikowi V w pierwszym kroku realizacji protokołu). Protokół wymiany klucza Diffiego–Hellmana przedstawiamy na rysunku 8.5.

- (1) U wybiera losowo  $a_U$ ,  $0 \leq a_U \leq p - 2$
- (2) U oblicza  $\alpha^{a_U} \bmod p$  i przekazuje wynik użytkownikowi V
- (3) V wybiera losowo  $a_V$ ,  $0 \leq a_V \leq p - 2$
- (4) V oblicza  $\alpha^{a_V} \bmod p$  i przekazuje wynik użytkownikowi U
- (5) U oblicza

$$K = (\alpha^{a_V})^{a_U} \bmod p,$$

a V oblicza

$$K = (\alpha^{a_U})^{a_V} \bmod p$$

RYSUNEK 8.5. Schemat wymiany kluczy Diffiego–Hellmana

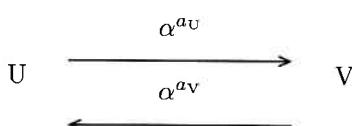
Po zakończeniu protokołu obaj użytkownicy, U i V, otrzymają w wyniku ten sam klucz:

$$K = \alpha^{a_U a_V} \bmod p.$$

Protokół, o którym mowa, jest bardzo podobny do wcześniej opisanego schematu wstępnej dystrybucji klucza Diffiego–Hellmana. Różnica polega na tym, że wykładniki  $a_U$  i  $a_V$  użytkowników U i V, odpowiednio, nie są stałe, lecz wybierane na nowo po każdym uruchomieniu protokołu. Ponadto w tym protokole zarówno U, jak i V mają pewność świeżości klucza, gdyż klucz sesji zależy od obu losowych wykładników  $a_U$  i  $a_V$ .

#### 8.4.1. Protokół wymiany klucza z uwierzytelnieniem

Protokół wymiany klucza Diffiego–Hellmana powinien przebiegać według następującego schematu:

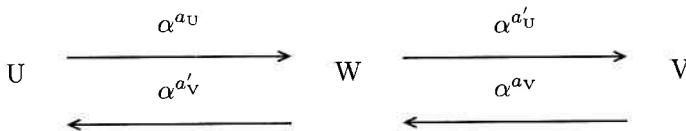


Niestety, protokół ten jest podatny na działania aktywnego przeciwnika stosującego atak typu *intruz-w-środku*. W *The Lucy Show*<sup>†</sup> był epizod, w którym

<sup>†</sup> *The Lucy Show* – serial emitowany w USA w latach sześćdziesiątych XX w. (przyp. tłum.).

Vivian Vance ma randkę w restauracji, je właśnie z partnerem kolację, a pod stołem siedzi ukryta Lucille Ball. Vivian i jej adorator chcą podać sobie ręce pod stołem, więc Lucy, aby uniknąć wpadki, sama trzyma każde z nich za rękę, tak by oboje sądzili, że ściskają się nawzajem.

Atak na protokół **wymiany klucza Diffiego–Hellmana** typu intruz-w-środku działa podobnie. Przeciwnik W przechwytuje wiadomości między U i V i podstawia za nie własne, tak jak to pokazuje następujący diagram:



W końcu w wyniku protokołu U w rzeczywistości uzgadnia z W wspólny klucz tajny  $\alpha^{a_U a'_V}$ , podczas gdy V uzgadnia z W klucz tajny  $\alpha^{a'_U a_V}$ . Gdy U szyfruje wiadomość wysyłaną do V, W potrafi ją odczytać, choć V tego zrobić nie potrafi (podobnie gdy V wysyła wiadomość do U).

Oczywiście, dla U i V pewność, że wymiana informacji odbywa się rzeczywiście między nimi, bez udziału W, jest sprawą kluczową. Przed wymianą kluczy mogliby oni zrealizować odrębny protokół służący do wzajemnego ustalenia swojej tożsamości, na przykład jeden ze schematów identyfikacji, które opiszymy w rozdziale 9. Nie chroni to jednak przed atakiem intruza, jeśli W zdecyduje się zachować bierność aż do momentu potwierdzenia tożsamości przez obu użytkowników U i V. Tak więc protokół uzgadniania klucza powinien sam uwierzytelniać tożsamość obu komunikujących się użytkowników i to jednocześnie z procesem ustalania klucza. Taki protokół nazwiemy *uwierzytelnionym uzgadnianiem klucza* (ang. *authenticated key agreement*).

Opiszemy tu protokół uwierzytelnionego uzgadniania klucza, który powstał przez modyfikację schematu **wymiany klucza Diffiego–Hellmana**. Protokół zakłada publiczną dostępność liczby pierwszej  $p$  i elementu pierwotnego  $\alpha$ , odwołuje się też do certyfikatów. Każdy użytkownik U dysponuje schematem podpisu z algorytmem weryfikacji  $ver_U$  i algorytmem podpisu  $sig_U$ ; także TA ma schemat podpisu z publicznym algorytmem weryfikacji  $ver_{TA}$ . Ponadto każdy użytkownik U ma certyfikat

$$\mathbf{C}(U) = (\text{ID}(U), ver_U, sig_{TA}(\text{ID}(U), ver_U)),$$

gdzie  $\text{ID}(U)$  jest informacją identyfikującą U.

Uwierzytelnione uzgadnianie klucza, czyli **wymiana klucza z uwierzytelnieniem** (ang. *station-to-station protocol*, STS), zawdzięczamy Diffiemu, Van Oorschotowi i Wienerowi. Protokół przedstawiony na rysunku 8.6 jest nieznacznie uproszczony; można go używać w sposób zgodny z protokołami ISO 9798-3.

- (1) użytkownik U wybiera losowo liczbę  $a_U$ ,  $0 \leq a_U \leq p - 2$   
 (2) U oblicza

$$\alpha^{a_U} \bmod p$$

i przekazuje wynik użytkownikowi V

- (3) V wybiera losowo liczbę  $a_V$ ,  $0 \leq a_V \leq p - 2$   
 (4) V oblicza

$$\alpha^{a_V} \bmod p,$$

a następnie

$$K = (\alpha^{a_U})^{a_V} \bmod p$$

oraz

$$y_V = \text{sig}_V(\alpha^{a_V}, \alpha^{a_U})$$

- (5) V przesyła  $(C(V), \alpha^{a_V}, y_V)$  do U  
 (6) U oblicza

$$K = (\alpha^{a_V})^{a_U} \bmod p;$$

weryfikuje  $y_V$  za pomocą  $\text{ver}_V$  oraz  $C(V)$  za pomocą  $\text{ver}_{TA}$

- (7) U oblicza

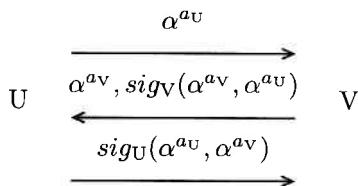
$$y_U = \text{sig}_U(\alpha^{a_U}, \alpha^{a_V})$$

i przesyła  $(C(U), y_U)$  do V

- (8) V weryfikuje  $y_U$  za pomocą  $\text{ver}_U$  oraz  $C(U)$  za pomocą  $\text{ver}_{TA}$

**RYSUNEK 8.6.** Uproszczony protokół wymiany klucza z uwierzytelnieniem

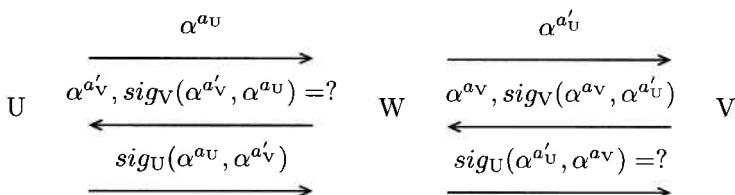
Wymiana informacji w uproszczonym protokole STS (z pominięciem certyfikatów) odbywa się według następującego schematu:



Zobaczmy, jak takie rozwiązanie chroni przed atakiem typu intruz-w-środku. Jak poprzednio, W przechwytuje informację  $\alpha^{a_U}$  i zastępuje ją przez  $\alpha^{a'_U}$ .

Dalej, W otrzymuje od V wartości  $\alpha^{av}$  i  $sig_V(\alpha^{av}, \alpha^{au})$ . Chciałby teraz, jak poprzednio, zastąpić  $\alpha^{av}$  przez  $\alpha^{a'v}$ , lecz tym razem musiałby zastąpić także  $sig_V(\alpha^{av}, \alpha^{au})$  przez  $sig_V(\alpha^{a'v}, \alpha^{au})$ . Niestety, nie potrafi obliczyć podpisu użytkownika V pod  $(\alpha^{a'v}, \alpha^{av})$ , ponieważ nie zna algorytmu podpisu  $sig_V$ . Podobnie W nie może zastąpić  $sig_U(\alpha^{av}, \alpha^{a'u})$  przez  $sig_U(\alpha^{a'u}, \alpha^{av})$ , ponieważ nie zna również algorytmu podpisu użytkownika U.

Ilustruje to następujący diagram:



Właśnie użycie podpisów uniemożliwia przeprowadzenie ataku typu intruz-w-środku.

Protokół opisany na rysunku 8.6 nie obejmuje potwierdzenia klucza. Jednak nietrudno go tak zmodyfikować, aby posiadał i tę cechę. Wystarczy określić

$$y_V = e_K(sig_V(\alpha^{av}, \alpha^{au}))$$

w kroku 4 oraz

$$y_U = e_K(sig_U(\alpha^{av}, \alpha^{a'u}))$$

w kroku 6. (Podobnie jak w systemie Kerberos potwierdzenie klucza uzyskujemy przez zaszyfrowanie znanej wielkości za pomocą nowego klucza sesji). Tak zmodyfikowany protokół nosi nazwę protokołu wymiany klucza z uwierzytelnieniem. Pozostawiamy zainteresowanemu Czytelnikowi zadanie uzupełnienia szczegółów.

#### 8.4.2. Protokół uzgadniania klucza MTI

Matsumoto, Takashima oraz Imai opracowali wiele interesujących protokołów uzgadniania klucza, modyfikując schemat wymiany klucza Diffiego–Hellmana. Protokoły te, które nazwiemy **protokołami MTI**, nie wymagają od użytkowników U i V obliczania podpisów. Są to protokoły *dwuprzebiegowe* (ang. *two-pass protocol*), gdyż odbywa się w nich tylko dwukrotne przekazanie informacji (jedno od U do V, drugie od V do U). Protokół STS należałoby w tym kontekście określić jako trójprzebiegowy.

Przedstawimy jeden z protokołów MTI oparty na tych samych założeniach, co schemat wymiany klucza Diffiego–Hellmana. Przyjmujemy, że liczba pierwsza  $p$  i element pierwotny  $\alpha$  są publicznie znane. Każdy użytkownik U dysponuje ciągiem identyfikacyjnym ID(U), tajnym wykładnikiem  $a_U$  ( $0 \leq a_U \leq p - 2$ ) oraz odpowiednią wartością publiczną

$$b_U = \alpha^{av} \bmod p.$$

Z kolei TA używa schematu podpisu z (publicznym) algorytmem weryfikacji  $ver_{TA}$  i tajnym algorytmem podpisu  $sig_{TA}$ .

Ponadto każdy użytkownik U posiada certyfikat

$$\mathbf{C}(U) = (\text{ID}(U), b_U, sig_{TA}(\text{ID}(U), b_U)),$$

w którym  $b_U$  powstaje w przedstawiony wyżej sposób.

Protokół uzgadniania klucza MTI jest przedstawiony na rysunku 8.7. Po zakończeniu protokołu użytkownicy U i V uzyskają w wyniku obliczeń ten sam klucz:

$$K = \alpha^{r_U a_V + r_V a_U} \bmod p.$$

(1) użytkownik U wybiera losowo  $r_U$ ,  $0 \leq r_U \leq p - 2$ , i oblicza

$$s_U = \alpha^{r_U} \bmod p$$

(2) U przesyła  $(\mathbf{C}(U), s_U)$  do użytkownika V

(3) V wybiera losowo  $r_V$ ,  $0 \leq r_V \leq p - 2$ , i oblicza

$$s_V = \alpha^{r_V} \bmod p$$

(4) V przesyła  $(\mathbf{C}(V), s_V)$  do U

(5) U oblicza

$$K = s_V^{a_U} b_V^{r_U} \bmod p,$$

gdzie wartość  $b_V$  pochodzi z  $\mathbf{C}(V)$ ; natomiast V oblicza

$$K = s_U^{a_V} b_U^{r_V} \bmod p,$$

gdzie wartość  $b_U$  pochodzi z  $\mathbf{C}(U)$

**RYSUNEK 8.7. Protokół uzgadniania klucza Matsumoto-Takashimy-Imaiego**

Popatrzmy na przykład ilustrujący działanie protokołu.

### Przykład 8.3

Załóżmy, że wartości  $p = 27803$  i  $\alpha = 5$  są publicznie znane. Przypuśćmy, że U wybiera  $a_U = 21131$ ; wówczas oblicza

$$b_U = 5^{21131} \bmod 27803 = 21420$$

i umieszcza tę liczbę w certyfikacie.

Przypuśćmy dalej, że V wybiera  $a_V = 17555$ , po czym oblicza

$$b_V = 5^{17555} \bmod 27803 = 17100$$

i wpisuje wynik do swojego certyfikatu.

Jeśli teraz U wybierze  $r_U = 169$ , to wyśle użytkownikowi V wartość

$$s_U = 5^{169} \bmod 27803 = 6268.$$

Jeśli z kolei V wybierze  $r_V = 23456$ , to wyśle do U wartość

$$s_V = 5^{23456} \bmod 27803 = 26759.$$

Teraz U może obliczyć klucz

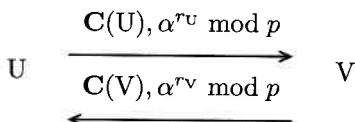
$$\begin{aligned} K_{U,V} &= s_V^{a_U} b_U^{r_U} \bmod p \\ &= 26759^{21131} 17100^{169} \bmod 27803 \\ &= 21600. \end{aligned}$$

Obliczenia V przebiegają w sposób następujący:

$$\begin{aligned} K_{U,V} &= s_U^{a_V} b_V^{r_V} \bmod p \\ &= 6268^{17555} 21420^{23456} \bmod 27803 \\ &= 21600. \end{aligned}$$

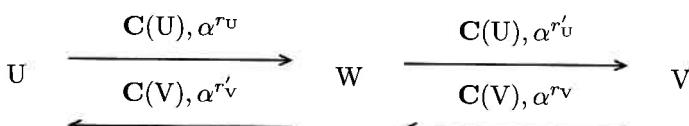
Tak więc U i V obliczyli ten sam klucz. □

Przekaz informacji w czasie realizacji protokołu ilustruje diagram:



Przyjrzyjmy się bezpieczeństwu tego schematu. Nietrudno wykazać, że bezpieczeństwo protokołu MTI w przypadku, gdy mamy do czynienia z biernym przeciwnikiem, jest dokładnie takie samo, jak dla problemu Diffiego–Hellmana – patrz ćwiczenia. Podobnie jak w przypadku wielu protokołów znacznie trudniej jest udowodnić bezpieczeństwo w razie ataku przeciwnika aktywnego, nie będziemy więc podejmować żadnych prób w tym kierunku, ograniczając się jedynie do nieformalnych rozważań.

Oto jedno z możliwych zagrożeń: wydawałoby się, że pominięcie podpisów w trakcie realizacji protokołu otwiera drogę dla ataków typu intruz-w-srodku. Rzeczywiście, może się zdarzyć, że W zmieni wartości przesyłane sobie nawzajem przez U i V. Jeden z możliwych scenariuszy mógłby wyglądać następująco:



W tej sytuacji U i V obliczą różne klucze. Użytkownik U obliczy

$$K = \alpha^{r_u a_v + r'_v a_u} \pmod{p},$$

podczas gdy V uzyska wynik

$$K = \alpha^{r'_u a_v + r v a_u} \pmod{p}.$$

Jednakże żadne z tych obliczeń nie może być wykonane przez W, ponieważ każde z nich wymaga znajomości tajnych wykładników  $a_U$  i  $a_V$ , odpowiednio. Tak więc jeśli nawet U i V otrzymają różne wartości klucza (co uczyni je bezużytecznymi dla obu), to żadnego z tych kluczy nie może obliczyć W (zakładając obliczeniową nieroziągalność problemu logarytmu dyskretnego). Inaczej mówiąc, każdy z użytkowników U i V ma pewność, że drugi z nich jest jedynym użytkownikiem sieci, który może obliczyć ten sam klucz, co on. Tę własność niekiedy nazywa się *pośrednim uwierzytelnieniem klucza* (ang. *implicit key authentication*).

#### 8.4.3. Uzgadnianie klucza z użyciem klucza samopotwierdzającego

Opiszymy tu zaproponowaną przez Giraulta metodę uzgadniania klucza, w której certyfikaty są niepotrzebne, gdyż wartość klucza publicznego oraz identyfikator użytkownika pośrednio uwierzytelniają się wzajemnie.

**Schemat Giraulta** łączy cechy kryptosystemu RSA oraz metody logarytmu dyskretnego. Niech  $n = pq$ , gdzie  $p = 2p_1 + 1$ ,  $q = 2q_1 + 1$ ; zakładamy, że  $p$ ,  $q$ ,  $p_1$ ,  $q_1$  są dużymi liczbami pierwszymi. Grupa multiplikatywna  $\mathbb{Z}_n^*$  jest izomorficzna z grupą  $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$ , zatem maksymalny rząd elementu grupy  $\mathbb{Z}_n^*$  jest równy najmniejszej wspólnej wielokrotności  $p - 1$  i  $q - 1$ , czyli  $2p_1q_1$ . Niech  $\alpha$  będzie elementem rzędu  $2p_1q_1$ . Wówczas podgrupa cykliczna grupy  $\mathbb{Z}_n^*$  generowana przez  $\alpha$  stanowi odpowiednie tło dla problemu logarytmu dyskretnego.

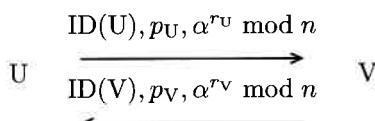
W schemacie Giraulta tylko TA zna rozkład liczby  $n$  na czynniki pierwsze. Wartości  $n$  i  $\alpha$  są publicznie dostępne, natomiast liczby  $p$ ,  $q$ ,  $p_1$  oraz  $q_1$  pozostają tajne. TA wybiera publiczny wykładnik szyfrowania RSA; nazwijmy go  $e$ . Odpowiadający mu wykładnik deszyfrowania  $d$  jest tajny (przypomnijmy, że  $d = e^{-1} \pmod{\phi(n)}$ ).

Jak w poprzednich schematach, tak i tu użytkownik U dysponuje ciągiem identyfikacyjnym  $ID(U)$ . Ponadto otrzymuje on od TA *samopotwierdzający klucz publiczny*  $p_U$ , jak to jest pokazane na rysunku 8.8. Zauważmy, że do tworzenia klucza  $p_U$  użytkownik U potrzebuje pomocy TA. Znając  $p_U$  i  $ID(U)$ , można obliczyć

$$b_U = p_U^e + ID(U) \pmod{n},$$

korzystając z ogólnie dostępnych informacji.

**Protokół uzgadniania klucza Giraulta** przedstawiamy na rysunku 8.9. Przekazywanie informacji w ramach protokołu ilustruje następujący diagram:



- (1) użytkownik U wybiera tajny wykładnik  $a_U$  i oblicza

$$b_U = \alpha^{a_U} \pmod{n}$$

- (2) U podaje TA wartości  $a_U$  i  $b_U$   
 (3) TA oblicza

$$p_U = (b_U - \text{ID}(U))^d \pmod{n}$$

- (4) TA podaje U wartość  $p_U$

**RYSUNEK 8.8.** Udział TA w tworzeniu samopotwierdzającego klucza publicznego

- (1) użytkownik U wybiera losowo  $r_U$  i oblicza

$$s_U = \alpha^{r_U} \pmod{n}$$

- (2) U przesyła  $\text{ID}(U)$ ,  $p_U$  oraz  $s_U$  do użytkownika V  
 (3) V wybiera losowo  $r_V$  i oblicza

$$s_V = \alpha^{r_V} \pmod{n}$$

- (4) V przesyła  $\text{ID}(V)$ ,  $p_V$  oraz  $s_V$  do U  
 (5) U oblicza

$$K = s_V^{a_U} (p_V^e + \text{ID}(V))^{r_U} \pmod{n},$$

natomiast V oblicza

$$K = s_U^{a_V} (p_U^e + \text{ID}(U))^{r_V} \pmod{n}$$

**RYSUNEK 8.9.** Protokół uzgadniania klucza Giraulta

Po zakończeniu protokołu każdy z użytkowników U i V uzyska ten sam obliczony przez siebie klucz

$$K = \alpha^{r_U a_V + r_V a_U} \pmod{n}.$$

Oto przykład wymiany kluczy za pomocą schematu Giraulta.

#### Przykład 8.4

Niech  $p = 839$  i  $q = 863$ . Wówczas  $n = 724057$  oraz  $\phi(n) = 722356$ . Element  $\alpha = 5$  ma rzad  $2p_1q_1 = \phi(n)/2$ . Przypuśćmy, że wybranym przez TA wykładnikiem deszyfrowania RSA jest  $d = 125777$ ; wtedy  $e = 84453$ .

Przyjmijmy, że  $ID(U) = 500021$  oraz  $a_U = 111899$ . Wówczas  $b_U = 488889$  oraz  $p_U = 650704$ . Założymy również, że  $ID(V) = 500022$  i  $a_V = 123456$ , co daje  $b_V = 111692$  i  $p_V = 683556$ .

Jeśli użytkownicy U i V chcą wymienić klucz i U wybierze  $r_U = 56381$  (co prowadzi do  $s_U = 171007$ ), a V wybierze  $r_V = 356935$  (co prowadzi do  $s_V = 320688$ ), to obaj obliczą ten sam klucz  $K = 42869$ .  $\square$

Zobaczmy, jak samopotwierdzające klucze chronią przed jednym określonym rodzajem ataku. Ponieważ TA nie podpisuje wartości  $b_U$ ,  $p_U$  oraz  $ID(U)$ , zatem nikt inny nie jest w stanie sprawdzić bezpośrednio ich autentyczności. Przypuśćmy, że W fałszuje te informacje (a więc nie są one wygenerowane w porozumieniu z TA), by podszyć się pod użytkownika U. Jeśli W zacznie od  $ID(U)$  i fałszywej wartości  $b'_U$ , to nie znajdzie sposobu na obliczenie wykładnika  $a'_U$ , odpowiadającego  $b'_U$  – o ile problem logarytmu dyskretnego jest nierozwiązalny obliczeniowo. Bez znajomości wykładnika  $a'_U$  przeciwnik W (udający U) nie będzie mógł obliczyć klucza.

Gdy W działa jako intruz-w-srodku, sytuacja jest podobna. Przeciwnik W może przeszkodzić użytkownikom U i V w uzgodnieniu wspólnego klucza, ale nie może powtórzyć obliczeń żadnego z nich. Tak więc omawiany schemat, podobnie jak protokół MTI, zapewnia pośrednie uwierzytelnienie klucza.

Uważny Czytelnik może zapytać, dlaczego oczekuje się od U, by przekazał TA wartość  $a_U$ . Rzeczywiście, TA może obliczyć  $p_U$  bezpośrednio z  $b_U$ , nie znając nawet  $a_U$ . Istotne jest tu jednak to, by TA, zanim obliczy dla użytkownika U wartość  $p_U$ , był przekonany, że U zna  $a_U$ .

Dla podkreślenia znaczenia tego założenia wykażemy, jak można zaatakować schemat, gdy TA rozdziela klucze publiczne  $p_U$  bez uprzedniego sprawdzenia, czy użytkownik zna wartość  $a_U$  odpowiadającą jego wartości  $b_U$ . Przypuśćmy, że W wybiera fałszywą wartość  $a'_U$  i dla niej oblicza

$$b'_U = \alpha^{a'_U} \pmod{n}.$$

Wówczas może ustalić odpowiedni klucz publiczny

$$p'_U = (b'_U - ID(U))^d \pmod{n}.$$

Istotnie, zacznie od obliczenia

$$b'_W = b'_U - ID(U) + ID(W),$$

po czym przekaże TA swoje dane:  $b'_W$  i  $ID(W)$ . Założymy, że TA przekazuje W publiczny klucz

$$p'_W = (b'_W - ID(W))^d \pmod{n}.$$

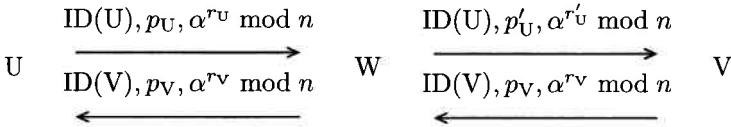
Z równości

$$b'_W - \text{ID}(W) \equiv b'_U - \text{ID}(U) \pmod{n}$$

wynika natychmiast, że

$$p'_W = p'_U.$$

Przypuśćmy teraz, że w późniejszym czasie użytkownicy U i V realizują protokół, natomiast W podstawia im swoją informację w sposób następujący:



Teraz V obliczy klucz

$$K' = \alpha^{r'_U a_V + r_V a'_U} \pmod{n},$$

podczas gdy obliczenia U dadzą wynik

$$K = \alpha^{r_U a_V + r_V a_U} \pmod{n}.$$

Jednocześnie W może obliczyć  $K'$  w inny sposób:

$$K' = s_V^{a'_U} (p_V^e + \text{ID}(V))^{r'_U} \pmod{n}.$$

W rezultacie W i V będą używać tego samego klucza, choć V pozostanie w przekonaniu, że dzieli klucz z U. W ten sposób W będzie mógł odczytywać wiadomości przekazywane U przez V.

## 8.5. Uwagi i bibliografia

Blom przedstawił swój schemat wstępnej dystrybucji klucza w [BL85]. Uogólnienia można znaleźć w pracach Blunda i innych [BDSHKVY93] oraz Beimela i Chora [BC94].

Diffie i Hellmann opublikowali swój algorytm wymiany klucza w [DH76]. Niezależnie pomysł wymiany klucza sformułował Merkle [ME78]. Informacje o uwierzytelnionej wymianie klucza pochodzą z pracy Diffiego, van Oorschota i Wienera [DVW92].

Opis wersji V systemu Kerberos można znaleźć w [KN93]. Schiller jest autorem opisowego artykułu o tym systemie [Sc94].

Protokoły Matsumoto, Takashimy i Imaiego można znaleźć w [MTI86]. Dysstrybucję kluczy samopotwierdzających zaproponował Girault [GIR91]. Schemat

przedstawiony przez niego był w istocie schematem wstępnej dystrybucji kluczowej; modyfikacja prowadząca do schematu uzgadniania klucza jest oparta na [RV94].

Dwie niedawne publikacje Rueppela i van Oorschota [RV94] oraz van Tilburga [vT93] zawierają przegląd wyników dotyczących dystrybucji i uzgadniania klucza.

## Ćwiczenia

- 8.1. Założymy, że stosujemy schemat Bloma z wartością  $k = 1$  dla grupy czterech użytkowników: U, V, W oraz X. Niech  $p = 7873$ ,  $r_U = 2365$ ,  $r_V = 6648$ ,  $r_W = 1837$  i  $r_X = 2186$ . Tajne wielomiany  $g$  są określone następująco:

$$g_U(x) = 6018 + 6351x$$

$$g_V(x) = 3749 + 7121x$$

$$g_W(x) = 7601 + 7802x$$

$$g_X(x) = 635 + 6828x$$

- (a) Oblicz klucz dla każdej pary użytkowników, upewniając się, czy obaj członkowie pary otrzymują ten sam klucz (a więc czy  $K_{U,V} = K_{V,U}$  itd.).  
 (b) Wskaż, jak W i X mogą wspólnie obliczyć  $K_{U,V}$ .

- 8.2. Założymy, że stosujemy schemat Bloma z wartością  $k = 2$  dla grupy pięciu użytkowników: U, V, W, X oraz Y. Niech  $p = 97$ ,  $r_U = 14$ ,  $r_V = 38$ ,  $r_W = 92$ ,  $r_X = 69$  i  $r_Y = 70$ . Tajne wielomiany  $g$  są określone następująco:

$$g_U(x) = 15 + 15x + 2x^2$$

$$g_V(x) = 95 + 77x + 83x^2$$

$$g_W(x) = 88 + 32x + 18x^2$$

$$g_X(x) = 62 + 91x + 59x^2$$

$$g_Y(x) = 10 + 82x + 52x^2$$

- (a) Pokaż, jak U i V, każdy z osobna, obliczą klucz  $K_{U,V} = K_{V,U}$ .  
 (b) Wskaż, jak W, X i Y mogą wspólnie obliczyć  $K_{U,V}$ .

- 8.3. Założymy, że U i V realizują schemat wymiany klucza Diffiego–Hellmanna z wartościami  $p = 27001$  i  $\alpha = 101$ . Przypuśćmy, że U wybiera  $a_U = 21768$ , a V wybiera  $a_V = 9898$ . Pokaż obliczenia, które wykonają U i V, i wskaż klucz, który będzie rezultatem ich obliczeń.

- 8.4. Założymy, że U i V realizują protokół MTI z wartościami  $p = 30113$  i  $\alpha = 52$ . Przypuśćmy, że U ma  $a_U = 8642$  i wybiera  $r_U = 28654$ , natomiast V ma  $a_V = 24673$  i wybiera  $r_V = 12385$ . Pokaż obliczenia, które wykonają U i V, i wskaż klucz, który będzie rezultatem ich obliczeń.

- 8.5. Bierny przeciwnik, próbujący obliczyć klucz wygenerowany przez U i V zgodnie z protokołem MTI, zetknie się ze szczególnym przypadkiem tego, co można nazwać problemem MTI, opisany na rysunku 8.10. Udosłownij, że każdy algorytm, który może posłużyć do rozwiązywania problemu MTI, może posłużyć także do rozwiązywania problemu Diffiego–Hellmanna i na odwrót.

**Dane**  $I = (p, \alpha, \beta, \gamma, \delta, \epsilon)$ , gdzie  $p$  jest liczbą pierwszą,  $\alpha \in \mathbb{Z}_p^*$  jest elementem pierwotnym oraz  $\beta, \gamma, \delta, \epsilon \in \mathbb{Z}_p^*$ .

**Cel** Obliczyć  $\beta^{\log_\alpha \gamma} \delta^{\log_\alpha \epsilon} \pmod{p}$ .

**RYSUNEK 8.10. Problem MTI**

- 8.6. Rozważmy schemat Giraulta z wartościami  $p = 167$ ,  $q = 179$ , a stąd  $n = 29893$ .  
Załóżmy, że  $\alpha = 2$  i  $e = 11101$ .

- Oblicz  $d$ .
- Przyjmując  $ID(U) = 10021$  i  $a_U = 9843$ , oblicz  $b_U$  i  $p_U$ . Podobnie, przyjmując  $ID(V) = 10022$  i  $a_V = 7692$ , oblicz  $b_V$  i  $p_V$ .
- Pokaż, jak obliczyć  $b_U$ , znając  $p_U$  oraz  $ID(U)$  i korzystając z publicznego wykładnika  $e$ . Podobnie pokaż, jak obliczyć  $b_V$ , znając wartości  $p_V$  i  $ID(V)$ .
- Przypuśćmy, że U wybiera  $r_U = 15556$ , a V wybiera  $r_V = 6420$ . Oblicz  $s_U$  oraz  $s_V$  i pokaż, jak każdy z użytkowników U i V obliczy wspólny klucz.

# 9

## *Schematy identyfikacji*

---

### **9.1. Wprowadzenie**

Metody kryptograficzneumożliwiają rozwiązywanie wielu pozornie nieroziąwalnych problemów. Jednym z nich jest skonstruowanie bezpiecznych schematów identyfikacji. W wielu codziennych sytuacjach zachodzi konieczność elektronicznego „udowodnienia” swojej tożsamości. Oto parę typowych przykładów:

- (1) Pobierając pieniądze z bankomatu, legitymujemy się kartą wraz z czterocyfrowym numerem identyfikacyjnym (PIN).
- (2) Płacąc kartą za zamówiony telefonicznie towar, podajemy numer karty kredytowej (i jej termin ważności).
- (3) Realizując międzynarodowe połączenie telefoniczne za pośrednictwem karty telefonicznej z przedpłatą, musimy znać numer telefonu kontaktowego i czterocyfrowy PIN.
- (4) Łącząc się ze zdalnym komputerem, podajemy aktualnie obowiązującą nazwę użytkownika i hasło.

W praktyce tego rodzaju schematy nie są implementowane w sposób wystarczająco bezpieczny. Protokoły realizowane podczas połączeń telefonicznych są narażone na podsłuch, a podsłuchujący przeciwnik może wykorzystać uzyskaną informację identyfikacyjną do własnych celów. Dotyczy to także odbiorcy informacji; wiele oszustw dotyczących kart płatniczych odbywa się w ten sposób. Karta bankomatowa jest nieco bezpieczniejsza, choć i tu występują słabe punkty. Na przykład, osoba monitorująca kanał komunikacyjny może zdobyć wszystkie informacje zakodowane na pasku magnetycznym, podobnie jak i sam PIN, co otwiera oszustowi drogę do rachunku bankowego posiadacza karty. Wreszcie, poważnym problemem jest bezpieczeństwo łączenia się ze zdalnym komputerem, gdyż ani nazwa użytkownika, ani jego hasło nie są w sieci szyfrowane. Każdy, kto monitoruje sieć, może uzyskać do nich dostęp.

Celem schematu identyfikacji jest zapobieganie takiej sytuacji, w której ktoś, kto podsłuchuje Alicję, gdy ta potwierdza swą tożsamość wobec Bolką, mógłby następnie podszyć się pod nią. Co więcej, schemat powinien również uniemożliwić Bolkowi podszycie się pod Alicję po potwierdzeniu przez nią swojej tożsamości wobec niego. Inaczej mówiąc, chodzi o to, by Alicja mogła udowodnić elektronicznie swoją tożsamość bez zdradzania swoich danych identyfikacyjnych.

Opracowano już wiele takich schematów identyfikacji. Z praktycznego punktu widzenia zadanie polega na znalezieniu schematu na tyle prostego, by można było go zrealizować na inteligentnej karcie, czyli zasadniczo na karcie kredytowej wyposażonej w układ scalony, który mógłby wykonać obliczenia arytmetyczne. Stąd zarówno ilość obliczeń, jak i wymagania pamięciowe muszą być ograniczone do niezbędnego minimum. Taka karta zapewniałaby większe bezpieczeństwo niż istniejące dziś karty bankomatowe. Warto jednak podkreślić, że mowa tu o zwiększym bezpieczeństwie wobec zagrożenia podsłuchem ze strony przeciwnika monitorującego kanał komunikacyjny. Sama karta „udowadnia” swoją tożsamość, zatem nie zyskujemy dodatkowej ochrony przed konsekwencjami zgubienia karty. Nadal PIN pozostaje niezbędnym elementem potwierdzania tożsamości osoby inicjującej proces identyfikacji.

Dalej w tym rozdziale opiszymy niektóre z częściej stosowanych schematów identyfikacji. Zaczniemy jednak od bardzo prostego schematu, który może być oparty na dowolnym systemie kryptograficznym z kluczem prywatnym, na przykład takim jak DES. Opisany na rysunku 9.1 protokół jest nazywany protokołem typu *pytanie-odpowiedź* (ang. *challenge-and-response*). Zakłada się w nim, że Alicja przedstawia się Bolkowi, przy czym oboje dzielą wspólny klucz tajny  $K$ , wyznaczający funkcję szyfrowania  $e_K$ .

- (1) Bolek wybiera pytanie  $x$  w postaci losowego ciągu 64 bitów i przekazuje je Alicji
- (2) Alicja oblicza

$$y = e_K(x)$$

i przekazuje wynik Bolkowi

- (3) Bolek oblicza

$$y' = e_K(x)$$

i sprawdza, że  $y' = y$

RYSUNEK 9.1. Protokół typu pytanie-odpowiedź

Przyjrzyjmy się działaniu takiego protokołu na przykładzie.

### Przykład 9.1

Przymijmy, że Alicja i Bolek używają funkcji szyfrowania wykonującej modułarne potęgowanie:

$$e_K(x) = x^{101379} \bmod 167653.$$

Jeśli Bolek poda wartość  $x = 77835$ , Alicja odpowie liczbą  $y = 100369$ .  $\square$

Praktycznie wszystkie schematy identyfikacji są protokołami typu pytanie–odpowiedź, choć te najbardziej użyteczne nie wymagają wspólnego klucza. Tym zajmiemy się w dalszej części rozdziału.

## 9.2. Schemat identyfikacji Schnorra

Zaczniemy od opisu **schematu identyfikacji Schnorra**, jednego z najciekawszych praktycznych schematów identyfikacyjnych. Zakładamy istnienie wiarygodnego czynnika, którego nazwiemy TA. TA wybierze parametry schematu:

- (1)  $p$  – dużą liczbę pierwszą (czyli  $p \geq 2^{512}$ ), taką że problem logarytmu dyskretnego jest obliczeniowo nieroziągalny w  $\mathbb{Z}_p^*$ ;
- (2)  $q$  – duży dzielnik pierwszy liczby  $p - 1$  (czyli  $q \geq 2^{140}$ );
- (3) liczbę  $\alpha \in \mathbb{Z}_p^*$  rzędu  $q$  (taką liczbę  $\alpha$  można obliczyć, podnosząc element pierwotny do potęgi  $p - 1$ );
- (4) parametr bezpieczeństwa  $t$ , taki że  $q > 2^t$ ; dla większości praktycznych zastosowań wartość  $t = 40$  zapewnia wystarczające bezpieczeństwo;
- (5) TA ustala ponadto bezpieczny schemat podpisu z tajnym algorytmem podpisu  $sig_{TA}$  i publicznym algorytmem weryfikacji  $ver_{TA}$ ;
- (6) bezpieczną funkcję skrótu (jak zwykle, informacja podlega skróceniu przed podpisaniem; dla ułatwienia śledzenia konstrukcji protokołu będziemy pomijać w opisach protokołów proces skracania).

Parametry  $p$ ,  $q$  i  $\alpha$  oraz publiczny algorytm weryfikacji  $ver_{TA}$  i funkcja skrótu są powszechnie dostępne.

TA wystawia Alicji certyfikat. Gdy Alicja chce taki certyfikat otrzymać, postępuje się w sposób opisany na rysunku 9.2. Kiedykolwiek Alicja zechce później udowodnić swoją tożsamość, na przykład wobec Bolka, wykonuje protokół przedstawiony na rysunku 9.3.

Jak już stwierdziliśmy,  $t$  jest parametrem bezpieczeństwa. Służy on do tego, by osoba podszywająca się pod Alicję, powiedzmy Olga, nie była w stanie

- (1) TA ustala tożsamość Alicji tradycyjnymi metodami, takimi jak metryka urodzenia, dowód osobisty itp.; następnie buduje ciąg ID(Alicja) zawierający informacje identyfikacyjne
- (2) Alicja potajemnie wybiera losowy wykładnik  $a$ , taki że  $0 \leq a \leq q-1$  i oblicza

$$v = \alpha^{-a} \pmod{p},$$

po czym przesyła wartość  $v$  do TA

- (3) TA generuje podpis

$$s = \text{sig}_{\text{TA}}(\text{ID}(\text{Alicja}), v)$$

do Alicji trafia certyfikat

$$\mathbf{C}(\text{Alicja}) = (\text{ID}(\text{Alicja}), v, s)$$

**RYSUNEK 9.2.** Wydawanie certyfikatu dla Alicji

- (1) Alicja wybiera losową liczbę  $k$ , taką że  $0 \leq k \leq q-1$  i oblicza
- $\gamma = \alpha^k \pmod{p}$
- (2) Alicja przekazuje Bolekowi swój certyfikat  $\mathbf{C}(\text{Alicja}) = (\text{ID}(\text{Alicja}), v, s)$  oraz wartość  $\gamma$
- (3) Bolek weryfikuje podpis TA, sprawdzając, czy  $\text{ver}_{\text{TA}}(\text{ID}(\text{Alicja}), v, s) = \text{tak}$
- (4) Bolek wybiera losową liczbę  $r$ , taką że  $1 \leq r \leq 2^t$  i przekazuje ją Alicji
- (5) Alicja oblicza

$$y = k + ar \pmod{q}$$

i przekazuje wynik  $y$  Bolekowi

- (6) Bolek sprawdza, że

$$\gamma \equiv \alpha^y v^r \pmod{p}$$

**RYSUNEK 9.3.** Schemat identyfikacji Schnorra

odgadnąć pytania Bolka, czyli wartości  $r$ . Gdyby tak się stało, Olga mogłaby wybrać dowolną wartość  $y$  i obliczyć

$$\gamma = \alpha^y v^r \pmod{p}.$$

Przekazałaby wtedy Bolkowi wartość  $\gamma$  w kroku 1. Później, po otrzymaniu pytania  $r$ , odpowiałaby wybraną przez siebie wcześniej wartością  $y$ . Wówczas Bolek uzyskałby potwierdzenie wartości  $\gamma$  w kroku 6.

Jeśli Bolek wybiera losowo liczbę  $r$ , prawdopodobieństwo jej poprawnego odgadnięcia przez Olgę jest równe  $2^{-t}$ , dlatego w większości przypadków wystarczy  $t = 40$ . Zwróćmy jednak uwagę na konieczność nowego, losowego wyboru  $r$  za każdym razem, gdy Alicja identyfikuje się wobec Bolka. Użycie tej samej wartości  $r$  umożliwiłoby Oldze podszycie się pod Alicję w sposób opisany wyżej.

W protokole weryfikacji dokonują się zasadniczo dwie rzeczy. Po pierwsze, podpis  $s$  potwierdza ważność certyfikatu Alicji. W ten sposób Bolek sprawdza podpis TA na certyfikacie Alicji, aby przekonać się o autentyczności samego certyfikatu. W istocie rzeczy tak samo używaliśmy certyfikatów w rozdziale 8.

Druga część protokołu dotyczy tajnej liczby  $a$ . Wartość  $a$  spełnia funkcję podobną do PIN, przekonuje bowiem Bolka, że osoba wykonująca protokół identyfikacji jest rzeczywiście Alicją. Jest jednak istotna różnica: w protokole identyfikacji wartość  $a$  nie jest ujawniana. Alicja (a dokładniej jej inteligentna karta) „udowadnia”, że zna wartość  $a$  w kroku 5 protokołu, obliczając  $y$  w odpowiedzi na pytanie  $r$  zadane przez Bolka. Ponieważ wartość  $a$  pozostaje tajna, tę technikę określa się jako *dowód przez wiedzę* (ang. *proof of knowledge*).

Za pomocą następujących kongruencji pokazujemy, że Alicja będzie w stanie udowodnić swoją tożsamość wobec Bolka:

$$\begin{aligned}\alpha^y v^r &\equiv \alpha^{k+ar} v^r \pmod{p} \\ &\equiv \alpha^{k+ar} \alpha^{-ar} \pmod{p} \\ &\equiv \alpha^k \pmod{p} \\ &\equiv \gamma \pmod{p}.\end{aligned}$$

Tak więc Bolek uzna dowód tożsamości Alicji (jeśli postępuje uczciwie). Mówimy wtedy, że protokół jest *zupełny*.

Oto mały przykład, ilustrujący procedurę pytania-odpowiedzi w ramach protokołu.

### Przykład 9.2

Niech  $p = 88667$ ,  $q = 1031$  oraz  $t = 10$ . Element  $\alpha = 70322$  ma w  $\mathbb{Z}_p^*$  rząd  $q$ . Założymy, że tajnym wykładnikiem Alicji jest liczba  $a = 755$ . Wtedy

$$\begin{aligned}v &= \alpha^{-a} \pmod{p} \\ &= 70322^{1031-755} \pmod{88667} \\ &= 13136.\end{aligned}$$

Przypuśćmy, że teraz Alicja wybiera  $k = 543$ . Oblicza wtedy

$$\begin{aligned}\gamma &= \alpha^k \pmod{p} \\ &= 70322^{543} \pmod{88667} \\ &= 84109\end{aligned}$$

i przesyła tę wartość  $\gamma$  Bolkowi. Gdy Bolek zada pytanie  $r = 1000$ , Alicja wykona następujące obliczenie:

$$\begin{aligned} y &= k + ar \bmod q \\ &= 543 + 755 \cdot 1000 \bmod 1031 \\ &= 851 \end{aligned}$$

i prześle Bolkowi tak uzyskaną wartość. Teraz Bolek sprawdza:

$$84109 \equiv 70322^{851} 13136^{1000} \pmod{88667}.$$

To pozwala mu uwierzyć, że po drugiej stronie kanału komunikacyjnego znajduje się Alicja.  $\square$

Spróbujmy teraz ustalić, w jaki sposób mógłby przeciwnik próbować wcielić się w Alicję. Oszust, powiedzmy Olga, mógłby starać się sfałszować certyfikat

$$\mathbf{C}'(\text{Alicja}) = (\text{ID}(\text{Alicja}), v', s'),$$

gdzie  $v' \neq v$ . Jednakże podpisem odpowiadającym  $(\text{ID}(\text{Alicja}), v')$  być powinno  $s'$ , co Bolek sprawdzi w kroku 3 protokołu. Jeśli schemat podpisu TA jest bezpieczny, Olga nie będzie umiała wygenerować podpisu  $s'$ , który zostałby później pozytywnie zweryfikowany przez Bolka.

Dalej, Olga mogłaby użyć poprawnego certyfikatu Alicji, czyli  $\mathbf{C}(\text{Alicja}) = (\text{ID}(\text{Alicja}), v, s)$  (przypomnijmy, że certyfikat nie jest tajny, a informacja w nim zawarta jest ujawniana przy każdej realizacji protokołu identyfikacji). Nie znając jednak wartości  $a$ , nie może udawać Alicji. Powodem jest „pytanie”  $r$  w kroku 4. W kroku 5 Olga musiałaby obliczyć  $y$ , ale  $y$  jest funkcją  $a$ . Obliczenie wartości  $a$  na podstawie znajomości  $v$  zakłada rozwiązanie problemu logarytmu dyskretnego, który – jak przyjmujemy – jest obliczeniowo nieroziwiążalny.

Dokładniej kwestię bezpieczeństwa protokołu ujmuje następujące twierdzenie.

### TWIERDZENIE 9.1

Jeśli Olga zna wartość  $\gamma$ , dla której prawdopodobieństwo jej skutecznego wcielenia się w Alicję w trakcie realizacji protokołu weryfikacji jest równe  $\epsilon \geq 1/2^{t-1}$ , to może ona obliczyć  $a$  w czasie wielomianowym.

**DOWÓD** W  $\epsilon$  przypadkach – spośród  $2^t$  możliwych pytań  $r$  – Olga może obliczyć wartość  $y$ , którą Bolek zaakceptuje w kroku 6 protokołu. Ponieważ z założenia  $\epsilon \geq 1/2^{t-1}$ , zatem  $2^t \epsilon \geq 2$ . Tak więc Olga może obliczyć wartości  $y_1, y_2, r_1$  oraz  $r_2$ , takie że

$$y_1 \not\equiv y_2 \pmod{q}$$

oraz

$$\gamma \equiv \alpha^{y_1} v^{r_1} \equiv \alpha^{y_2} v^{r_2} \pmod{p}.$$

Wynika stąd, że

$$\alpha^{y_1-y_2} \equiv v^{r_2-r_1} \pmod{p}.$$

Z kolei z równości  $v = \alpha^{-a}$  wnioskujemy, że

$$y_1 - y_2 \equiv a(r_1 - r_2) \pmod{q}.$$

Ponadto  $0 < |r_2 - r_1| < 2^t$ , a liczba  $q > 2^t$  jest pierwsza. W rezultacie  $\text{NWD}(r_2 - r_1, q) = 1$  i Olga może obliczyć

$$a = (y_1 - y_2)(r_1 - r_2)^{-1} \pmod{q},$$

zgodnie z tezą twierdzenia. ■

Z powyższego twierdzenia wynika, że ktoś, kto ma niezaniedbywalną szansę skutecznej realizacji protokołu identyfikacji, musi znać (lub umieć obliczyć w czasie wielomianowym) tajny wykładownik  $a$  Alicji. Tę własność określa się często jako *solidność* (ang. *soundness*).

Popatrzmy na przykład.

### Przykład 9.3

Przyjmijmy te same wartości parametrów co w przykładzie 9.2:  $p = 88667$ ,  $q = 1031$ ,  $t = 10$ ,  $\alpha = 70322$ ,  $a = 755$  i  $v = 13136$ . Założymy, że Olga dowiaduje się, iż

$$\alpha^{851}v^{1000} \equiv \alpha^{454}v^{19} \pmod{p}.$$

Wówczas może ona obliczyć

$$a = (851 - 454)(1000 - 19)^{-1} \pmod{1031} = 755$$

i w ten sposób odkryć tajny wykładownik Alicji. □

Wykazaliśmy, że protokół jest solidny i zupełny. Solidność i zupełność nie wystarczą jednak do zapewnienia bezpieczeństwa. Na przykład, gdyby Alicja po prostu ujawniła swój tajny wykładownik, powiedzmy Oldze, protokół pozostawałby solidny i zupełny – choć całkowicie nieodporny na ataki, jako że Olga mogłaby bez trudu wcielać się w Alicję.

Taka sytuacja zmusza do rozważenia kwestii tajnych informacji, które ujawnia się czynnikowi weryfikującemu (lub obserwatorowi), uczestniczącemu w realizacji protokołu (w omawianym tu protokole tajną informacją jest wartość wykładownika  $a$ ). Przystępujemy do działania w nadziei, że w trakcie udowadniania swojej tożsamości przez Alicję Olga nie potrafi zdobyć żadnej informacji o wykładowniku  $a$ ; w przeciwnym razie mogłaby skutecznie się pod nią podszyć.

Ogólniej, możemy wyobrazić sobie sytuację, w której Alicja przy różnych okazjach potwierdza swoją tożsamość wobec Olgi. Być może Olga nie wybiera

swoich pytań (czyli wartości  $r$ ) losowo. Po wielu przebiegach protokołu Olga może próbować ustalić wartość  $a$ , która umożliwia jej podszywanie się pod Alicję. Jeśli Olga, uczestnicząc w wielomianowej liczbie realizacji protokołu i wykonując wielomianową liczbę obliczeń, nie jest w stanie zdobyć jakiekolwiek informacji o  $a$ , możemy uznać, że protokół jest bezpieczny.

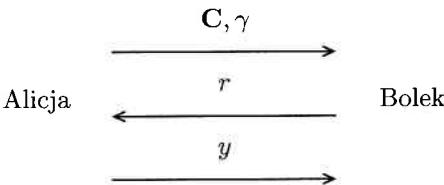
Nie udowodniono dotąd, że schemat Schnorra jest bezpieczny. W następnym podrozdziale pokażemy jego modyfikację wprowadzoną przez Okamoto, dla której taki dowód można przeprowadzić przy pewnym założeniu obliczeniowym.

Schemat Schnorra zaprojektowano tak, by był szybki i skuteczny, zarówno z punktu widzenia obliczeniowego, jak i pod względem ilości informacji wymienianej w trakcie realizacji protokołu; jednocześnie ma on minimalizować ilość obliczeń, które musi wykonać Alicja. Jest to pożądane choćby dlatego, że w wielu praktycznych zastosowaniach obliczenia Alicji będą wykonywane przez inteligentną kartę o małej mocy obliczeniowej, podczas gdy Bolek będzie przeprowadzał swoje obliczenia na znacznie potężniejszym komputerze.

Dla naszych rozważań przyjmijmy, że  $\text{ID}(\text{Alicja})$  jest ciągiem 512-bitowym;  $v$  będzie również zawierało 512 bitów, natomiast  $s$  będzie ciągiem 320-bitowym, jeśli schematem podpisu jest DSS. Wtedy pełny certyfikat  $C(\text{Alicja})$  (który musi być przechowywany na inteligentnej karcie Alicji) będzie składał się z 1344 bitów.

Przyjrzyjmy się obliczeniom Alicji. Krok 1 wymaga modularnego potęgowania. W kroku 5 jest wykonywane jedno modularne dodawanie i jedno modularne mnożenie. Z obliczeniowego punktu widzenia najtrudniejsze jest potęgowanie, lecz to działanie można, jeśli się chce, wykonać wcześniej off-line (poza siecią). Wówczas obliczenia, które Alicja musi wykonać on-line, są już bardzo ograniczone.

Nietrudno także obliczyć liczbę bitów, które trzeba przekazać podczas przebiegu protokołu. Obieg informacji można przedstawić w postaci diagramu:



W kroku 2 Alicja przekazuje Bolekowi  $1344 + 512 = 1856$  bitów informacji, Bolek przesyła Alicji 40 bitów w kroku 4, wreszcie w kroku 6 Alicja dostarcza Bolekowi 140 bitów. Jak widać, wymagania komunikacyjne są także bardzo skromne.

### 9.3. Schemat identyfikacji Okamoto

Przedstawimy tu pewną modyfikację schematu Schnorra autorstwa Okamoto. Można udowodnić jej bezpieczeństwo, jeśli założy się, że pewien określony algorytm dyskretny jest obliczeniowo nieroziągalny w  $\mathbb{Z}_p$ .

Na początek TA wybiera  $p$  i  $q$  tak jak w schemacie Schnorra; wybiera ponadto dwa elementy  $\alpha_1, \alpha_2 \in \mathbb{Z}_p^*$ , oba rzędu  $q$ . Wartość  $c = \log_{\alpha_1} \alpha_2$  pozostaje nieznana dla wszystkich uczestników, łącznie z Alicją. Przyjmujemy, że nikt (nawet koalicja, powiedzmy, Alicji z Olgą) nie jest w stanie obliczyć wartości  $c$ . Jak poprzednio TA wybiera schemat podpisu i funkcję skrótu. Certyfikat, który TA wydaje dla Alicji, powstaje w sposób opisany na rysunku 9.4. Na rysunku 9.5 przedstawiamy schemat identyfikacji Okamoto.

(1) TA ustala tożsamość Alicji i wystawia jej ciąg identyfikacyjny  $\text{ID}(\text{Alicja})$

(2) Alicja potajemnie wybiera dwa losowe wykładniki  $a_1, a_2$ , takie że  $0 \leq a_1, a_2 \leq q - 1$  i oblicza

$$v = \alpha_1^{-a_1} \alpha_2^{-a_2} \pmod{p},$$

po czym przekazuje TA wynik  $v$

(3) TA generuje podpis

$$s = \text{sig}_{\text{TA}}(\text{ID}(\text{Alicja}), v);$$

do Alicji trafia certyfikat

$$\mathbf{C}(\text{Alicja}) = (\text{ID}(\text{Alicja}), v, s)$$

#### RYSUNEK 9.4. Wydawanie Alicji certyfikatu

Oto przykład zastosowania tego schematu.

#### Przykład 9.4

Jak w poprzednich przykładach przyjmiemy  $p = 88667$ ,  $q = 1031$  oraz  $t = 10$ . Niech  $\alpha_1 = 58902$  i  $\alpha_2 = 73611$  (obie te liczby mają rząd  $q$  w  $\mathbb{Z}_p^*$ ). Założymy teraz, że  $a_1 = 846$  i  $a_2 = 515$ ; wtedy  $v = 13078$ .

Przypuśćmy, że Alicja wybiera  $k_1 = 899$  i  $k_2 = 16$ ; wtedy  $\gamma = 14574$ . Jeśli Bolek wyśle pytanie  $r = 489$ , Alicja odpowie wartościami  $y_1 = 131$  i  $y_2 = 287$ . Bolek sprawdzi, że

$$58902^{131} 73611^{287} 13078^{489} \equiv 14574 \pmod{88667}$$

i w konsekwencji uzna dowód tożsamości Alicji. □

Dowód zupełności protokołu (czyli tego, że Bolek uzna tożsamość Alicji) jest bardzo prosty. Zasadnicza różnica między schematami Schnorra i Okamoto polega na tym, że dla tego drugiego można udowodnić jego bezpieczeństwo,

- (1) Alicja wybiera losowe liczby  $k_1, k_2$ , takie że  $0 \leq k_1, k_2 \leq q - 1$  i oblicza

$$\gamma = \alpha_1^{k_1} \alpha_2^{k_2} \pmod{p}$$

- (2) Alicja przekazuje Bolkowi swój certyfikat  $\mathbf{C}(\text{Alicja}) = (\text{ID}(\text{Alicja}), v, s)$  oraz wartość  $\gamma$
- (3) Bolek weryfikuje podpis TA, sprawdzając, czy  $\text{ver}_{\text{TA}}(\text{ID}(\text{Alicja}), v, s) = \text{tak}$
- (4) Bolek wybiera losową liczbę  $r$ , taką że  $1 \leq r \leq 2^t$  i przekazuje ją Alicji
- (5) Alicja oblicza

$$y_1 = k_1 + a_1 r \pmod{q}$$

oraz

$$y_2 = k_2 + a_2 r \pmod{q}$$

i przekazuje Bolkowi wartości  $y_1$  i  $y_2$

- (6) Bolek sprawdza, że

$$\gamma \equiv \alpha_1^{y_1} \alpha_2^{y_2} v^r \pmod{p}$$

### RYSUNEK 9.5. Schemat identyfikacji Okamoto

pod warunkiem że możemy uznać obliczenie logarytmu dyskretnego  $\log_{\alpha_1} \alpha_2$  za praktycznie niewykonalne.

Dowód bezpieczeństwa jest dość subtelny. Oto szkic rozumowania.

Jak poprzednio Alicja, realizując protokół, przedstawia się Oldze wielomianowo wiele razy. Przyjmijmy (w nadzieję na doprowadzenie do sprzeczności), że Olga jest w stanie uzyskać informację o wartości tajnych wykładowników Alicji, a więc liczb  $a_1$  i  $a_2$ . Jeśli tak, to – jak wykażemy – z dużym prawdopodobieństwem Alicja i Olga potrafią razem obliczyć logarytm dyskretny  $c$  w czasie wielomianowym – co stoi w sprzeczności z przyjętym wyżej założeniem i dowodzi, że uczestnictwo Olgi w protokole nie daje jej możliwości zdobycia informacji o wykładownikach Alicji.

Pierwsza część tej procedury przypomina dowód solidności schematu Schnorra.

### TWIERDZENIE 9.2

Jeśli Olga zna wartość  $\gamma$ , dla której prawdopodobieństwo jej skutecznego powiadania się za Alicję w trakcie realizacji protokołu weryfikacji jest równe  $\epsilon \geq 1/2^{t-1}$ , to może ona obliczyć w czasie wielomianowym wartości  $b_1$  i  $b_2$ , takie że

$$v \equiv \alpha_1^{-b_1} \alpha_2^{-b_2} \pmod{p}.$$

**DOWÓD** W  $\epsilon$  przypadkach – spośród  $2^t$  możliwych pytań  $r$  – Olga może obliczyć wartości  $y_1, y_2, z_1, z_2, r$  i  $s$ , takie że  $r \neq s$  oraz

$$\gamma \equiv \alpha_1^{y_1} \alpha_2^{y_2} v^r \equiv \alpha_1^{z_1} \alpha_2^{z_2} v^s \pmod{p}.$$

Niech

$$b_1 = (y_1 - z_1)(r - s)^{-1} \pmod{q}$$

oraz

$$b_2 = (y_2 - z_2)(r - s)^{-1} \pmod{q}.$$

Łatwo wówczas sprawdzić, że

$$v \equiv \alpha_1^{-b_1} \alpha_2^{-b_2} \pmod{p},$$

zgodnie z tezą twierdzenia. ■

Zobaczmy teraz, jak Alicja razem z Olgą mogą obliczyć wartość  $c$ .

### TWIERDZENIE 9.3

Jeśli Olga zna wartość  $\gamma$ , dla której prawdopodobieństwo jej skutecznego podawania się za Alicję w trakcie realizacji protokołu weryfikacji jest równe  $\epsilon \geq 1/2^{t-1}$ , to z prawdopodobieństwem  $1 - 1/q$  może ona razem z Alicją obliczyć  $\log_{\alpha_1} \alpha_2$  w czasie wielomianowym.

**DOWÓD** Jak wynika z twierdzenia 9.2, Olga może ustalić wartości  $b_1$  i  $b_2$ , takie że

$$v \equiv \alpha_1^{-b_1} \alpha_2^{-b_2} \pmod{p}.$$

Przypuśćmy teraz, że Alicja ujawnia Oldze wartości  $a_1$  i  $a_2$ . Oczywiście

$$v \equiv \alpha_1^{-a_1} \alpha_2^{-a_2} \pmod{p},$$

zatem

$$\alpha_1^{a_1-b_1} \equiv \alpha_2^{b_2-a_2} \pmod{p}.$$

Załóżmy, że  $(a_1, a_2) \neq (b_1, b_2)$ . Wówczas istnieje  $(a_2 - b_2)^{-1} \pmod{q}$ , a logarytm dyskretny

$$c = \log_{\alpha_1} \alpha_2 = (a_1 - b_1)(b_2 - a_2)^{-1} \pmod{q}$$

można obliczyć w czasie wielomianowym.

Pozostaje do rozważenia przypadek, gdy  $(a_1, a_2) = (b_1, b_2)$ . Wtedy wartości  $c$  nie można obliczyć w sposób opisany wyżej; wykażemy jednak, że taki

przypadek zdarzy się z bardzo małym prawdopodobieństwem  $1/q$ , a więc procedura obliczania  $c$ , którą stosują Alicja i Olga, prawie na pewno zakończy się powodzeniem.

Określmy

$$\mathcal{A} = \{(a'_1, a'_2) \in \mathbb{Z}_q \times \mathbb{Z}_q : \alpha_1^{-a'_1} \alpha_2^{-a'_2} \equiv \alpha_1^{-a_1} \alpha_2^{-a_2} \pmod{p}\}.$$

Zbiór  $\mathcal{A}$  składa się ze wszystkich par uporządkowanych, których elementy mogłyby być tajnymi wykładownikami Alicji. Zauważmy, że

$$\mathcal{A} = \{(a_1 - c\theta, a_2 + \theta) : \theta \in \mathbb{Z}_q\},$$

gdzie  $c = \log_{\alpha_1} \alpha_2$ . Tak więc  $\mathcal{A}$  zawiera  $q$  par uporządkowanych.

Obliczona przez Olgę para  $(b_1, b_2)$  z pewnością należy do zbioru  $\mathcal{A}$ . Wykażemy, że para ta jest niezależna od pary tajnych wykładowników Alicji  $(a_1, a_2)$ . Ponieważ para  $(a_1, a_2)$  została przez Alicję wybrana losowo, prawdopodobieństwo równości  $(a_1, a_2) = (b_1, b_2)$  musi być równe  $1/q$ .

Musimy zatem wyjaśnić, co rozumiemy przez „niezależność” pary  $(b_1, b_2)$  od pary  $(a_1, a_2)$ . Otóż para Alicji  $(a_1, a_2)$  jest jedną z  $q$  możliwych par uporządkowanych w zbiorze  $\mathcal{A}$ ; podczas identyfikacji Alicja nie dostarcza Oldze żadnych informacji o tym, która para jest „poprawna”. Mówiąc nieformalnie, Olga wie, że jedna z par ze zbioru  $\mathcal{A}$  zawiera wykładowniki Alicji, nie wie jednak która.

Przyjrzyjmy się informacji wymienianej w trakcie realizacji protokołu identyfikacji. Przy każdym wykonaniu protokołu Alicja wybiera wartość  $\gamma$ , natomiast Olga wybiera  $r$ . Dalej, Alicja podaje wartości  $y_1$  i  $y_2$ , takie że

$$\gamma \equiv \alpha_1^{y_1} \alpha_2^{y_2} v^r \pmod{p}.$$

Przypomnijmy, że Alicja oblicza

$$y_1 = k_1 + a_1 r \pmod{q}$$

oraz

$$y_2 = k_2 + a_2 r \pmod{q},$$

gdzie

$$\gamma = \alpha_1^{k_1} \alpha_2^{k_2} \pmod{p}.$$

Zauważmy jednak, że wartości  $k_1$  i  $k_2$  (podobnie jak  $a_1$  i  $a_2$ ) pozostają tajne.

Konkretna czwórka  $(\gamma, r, y_1, y_2)$ , generowana w trakcie jednego przebiegu protokołu, wydaje się zależeć od wartości w parze uporządkowanej  $(a_1, a_2)$  Alicji, gdyż używamy tych wartości przy obliczaniu  $y_1$  i  $y_2$ . Udowodnimy jednak, że każdą taką czwórkę można równie dobrze wygenerować za pomocą dowolnej innej pary uporządkowanej  $(a'_1, a'_2) \in \mathcal{A}$ . Założymy bowiem, że  $(a'_1, a'_2) \in \mathcal{A}$ , a więc

$a'_1 = a_1 - c\theta$  oraz  $a'_2 = a_2 + \theta$ , gdzie  $0 \leq \theta \leq q - 1$ . Możemy wtedy przedstawić  $y_1$  i  $y_2$  w następujący sposób:

$$\begin{aligned} y_1 &= k_1 + a_1 r \\ &= k_1 + (a'_1 + c\theta)r \\ &= (k_1 + rc\theta) + a'_1 r \end{aligned}$$

oraz

$$\begin{aligned} y_2 &= k_2 + a_2 r \\ &= k_2 + (a'_2 - \theta)r \\ &= (k_2 - r\theta) + a'_2 r, \end{aligned}$$

gdzie wszystkie obliczenia są wykonywane w  $\mathbb{Z}_q$ . Inaczej mówiąc, czwórka  $(\gamma, r, y_1, y_2)$  jest zgodna również z parą uporządkowaną  $(a'_1, a'_2)$  przy losowym wyborze  $k'_1 = k_1 + rc\theta$  i  $k'_2 = k_2 - r\theta$  do obliczenia (tej samej) wartości  $\gamma$ . Stwierdziliśmy wyżej, że Alicja nie ujawnia wartości  $k_1$  i  $k_2$ , więc czwórka  $(\gamma, r, y_1, y_2)$  nie dostarcza informacji na temat żadnej pary ze zbioru  $\mathcal{A}$ . To kończy dowód. ■

Przedstawiony tu dowód bezpieczeństwa jest z pewnością elegancki i subtelny. Warto podsumować te własności protokołu, które umożliwiają przeprowadzenie takiego dowodu. Zasadniczy pomysł polega na wyborze dwóch tajnych wykładowników zamiast jednego. W zbiorze  $\mathcal{A}$  znajduje się  $q$  par uporządkowanych „równoważnych” parze Alicji  $(a_1, a_2)$ . Założeniem prowadzącym w końcu do sprzeczności jest przyjęcie, że znajomość dwóch różnych par ze zbioru  $\mathcal{A}$  pozwala efektywnie obliczyć logarytm dyskretny  $c$ . Alicja zna, rzecz jasna, jedną parę z  $\mathcal{A}$ ; wykazaliśmy ponadto, że jeśli Olga może podawać się za Alicję, to może obliczyć parę należącą do  $\mathcal{A}$  z dużym prawdopodobieństwem różną od pary Alicji. Tak więc Alicja i Olga razem mogą poznać dwie pary ze zbioru  $\mathcal{A}$  i w rezultacie obliczyć  $c$  – co jest właśnie poszukiwaną sprzecznością.

Oto przykład ilustrujący obliczanie wartości  $\log_{\alpha_1} \alpha_2$  przez Alicję i Olgę.

### Przykład 9.5

Jak w przykładzie 9.4 przyjmijmy  $p = 88667$ ,  $q = 1031$  oraz  $t = 10$  i założmy, że  $v = 13078$ .

Przypuśćmy, że Olga sprawdziła, że

$$\alpha_1^{131} \alpha_2^{287} v^{489} \equiv \alpha_1^{890} \alpha_2^{303} v^{199} \pmod{p}.$$

To pozwala jej obliczyć

$$b_1 = (131 - 890)(489 - 199)^{-1} \pmod{1031} = 456$$

oraz

$$b_2 = (287 - 303)(489 - 199)^{-1} \pmod{1031} = 519.$$

Teraz, używając dostarczonych przez Alicję wartości  $a_1$  i  $a_2$ , można obliczyć

$$c = (846 - 456)(519 - 515)^{-1} \bmod 1031 = 613.$$

Wartość  $c$  to w istocie  $\log_{\alpha_1} \alpha_2$ , co łatwo sprawdzić, obliczając

$$58902^{613} \bmod 88667 = 73611.$$

□

Warto na koniec podkreślić, że mimo braku znanego dowodu bezpieczeństwa schematu Schnorra (nawet przy założeniu praktycznej nierozerwialności problemu logarytmu dyskretnego), nie wskazano dotąd żadnego słabego punktu tego schematu. Co więcej, w praktyce schemat Schnorra może być preferowany w stosunku do schematu Okamoto po prostu dlatego, że jest szybszy.

## 9.4. Schemat identyfikacji Guillou–Quisquatera

W tym podrozdziale opiszemy inny schemat identyfikacji opracowany przez Guillou i Quisquatera i oparty na RSA.

Przygotowanie do realizacji schematu zaczyna się od tego, że TA wybiera dwie liczby pierwsze  $p$  oraz  $q$  i oblicza ich iloczyn  $n = pq$ . Wartości  $p$  i  $q$  są tajne, natomiast publicznie dostępny jest iloczyn  $n$ . Jak zwykle,  $p$  i  $q$  powinny być na tyle duże, by zadanie rozkładu na czynniki pierwsze liczby  $n$  było obliczeniowo niewykonalne. Ponadto, TA wybiera dużą liczbę pierwszą  $b$ , która posłuży za parametr bezpieczeństwa, a także jako publiczny wykładnik szyfrowania RSA; przyjmijmy, że  $b$  jest liczbą 40-bitową. Na koniec TA wybiera schemat podpisu oraz funkcję skrótu.

Certyfikat przekazywany Alicji przez TA powstaje zgodnie z opisem na rysunku 9.6. Gdy Alicja chce potwierdzić swoją tożsamość wobec, powiedzmy, Bolka, wykonuje protokół z rysunku 9.7. Wykażemy, że **schemat Guillou–Quisquatera** jest solidny i zupełny. Dotąd nie udowodniono jednak, że jest także bezpieczny (nawet przy założeniu bezpieczeństwa systemu kryptograficznego RSA).

### Przykład 9.6

Załóżmy, że TA wybiera  $p = 467$  i  $q = 479$ , a więc  $n = 223693$ . Przyjmijmy też, że  $b = 503$ , a tajna liczba Alicji  $u = 101576$ . Wówczas jej obliczenie będzie wyglądało tak:

$$\begin{aligned} v &= (u^{-1})^b \bmod n \\ &= (101576^{-1})^{503} \bmod 223693 \\ &= 89888. \end{aligned}$$

- (1) TA ustala tożsamość Alicji i wystawia jej ciąg identyfikacyjny  $\text{ID}(\text{Alicja})$
- (2) Alicja potajemnie wybiera liczbę całkowitą  $u$ , taką że  $0 \leq u \leq n-1$  i oblicza

$$v = (u^{-1})^b \bmod n,$$

po czym przekazuje TA wynik  $v$

- (3) TA generuje podpis

$$s = \text{sig}_{\text{TA}}(\text{ID}(\text{Alicja}), v);$$

do Alicji trafia certyfikat

$$\mathbf{C}(\text{Alicja}) = (\text{ID}(\text{Alicja}), v, s)$$

**RYSUNEK 9.6. Wydawanie Alicji certyfikatu**

- (1) Alicja wybiera losową liczbę  $k$ , taką że  $0 \leq k \leq n-1$  i oblicza
- $$\gamma = k^b \bmod n$$
- (2) Alicja przekazuje Bolkowi swój certyfikat  $\mathbf{C}(\text{Alicja}) = (\text{ID}(\text{Alicja}), v, s)$  oraz wartość  $\gamma$
  - (3) Bolek weryfikuje podpis TA, sprawdzając, czy  $\text{ver}_{\text{TA}}(\text{ID}(\text{Alicja}), v, s) = \text{tak}$
  - (4) Bolek wybiera losową liczbę  $r$ , taką że  $0 \leq r \leq b-1$ , i przekazuje ją Alicji
  - (5) Alicja oblicza

$$y = ku^r \bmod n$$

i przekazuje Bolkowi wynik  $y$

- (6) Bolek sprawdza, że

$$\gamma \equiv v^r y^b \pmod{n}$$

**RYSUNEK 9.7. Schemat identyfikacji Guillou–Quisquatera**

Przypuśćmy, że Alicja potwierdza swoją tożsamość wobec Bolka i wybiera  $k = 187485$ ; podaje wtedy Bolkowi wartość

$$\begin{aligned} \gamma &= k^b \bmod n \\ &= 187485^{503} \bmod 223693 \\ &= 24412. \end{aligned}$$

Jeśli Bolek odpowie pytaniem  $r = 375$ , Alicja obliczy

$$\begin{aligned} y &= ku^r \pmod{n} \\ &= 187485 \cdot 101576^{375} \pmod{223693} \\ &= 93725 \end{aligned}$$

i poda tę wartość Bolkowi. Bolek sprawdzi wówczas, że

$$24412 \equiv 89888^{375} 93725^{503} \pmod{223693}.$$

W ten sposób Bolek zaakceptuje dowód tożsamości Alicji.  $\square$

Jak zwykle dowód zupełności jest dość prosty:

$$\begin{aligned} v^r y^b &\equiv (u^{-b})^r (ku^r)^b \pmod{n} \\ &\equiv u^{-br} k^b u^{br} \pmod{n} \\ &\equiv k^b \pmod{n} \\ &\equiv \gamma \pmod{n}. \end{aligned}$$

Zajmijmy się teraz solidnością. Udowodnimy, że schemat jest solidny, pod warunkiem że nie można praktycznie obliczyć wartości  $u$ , znając  $v$ . Takie założenie wydaje się rozsądne, gdyż  $v$  powstaje z  $u$  w wyniku szyfrowania RSA.

#### TWIERDZENIE 9.4

Jeśli Olga zna wartość  $\gamma$ , dla której prawdopodobieństwo jej skutecznego podawania się za Alicję w trakcie realizacji protokołu weryfikacji jest równe  $\epsilon \geq 1/b$ , to może ona obliczyć  $u$  w czasie wielomianowym.

**DOWÓD** Dla pewnego  $\gamma$  Olga może obliczyć wartości  $y_1, y_2, r_1, r_2$ , takie że  $r_1 \neq r_2$  oraz

$$\gamma \equiv v^{r_1} y_1^b \equiv v^{r_2} y_2^b \pmod{n}.$$

Przypuśćmy, bez utraty ogólności, że  $r_1 > r_2$ . Wtedy

$$v^{r_1 - r_2} \equiv (y_2/y_1)^b \pmod{n}.$$

Ponieważ  $0 < r_1 - r_2 < b$  i  $b$  jest liczbą pierwszą, więc  $t = (r_1 - r_2)^{-1} \pmod{b}$  istnieje i Olga może tę wartość obliczyć w czasie wielomianowym za pomocą algorytmu Euklidesa. Mamy zatem

$$v^{(r_1 - r_2)t} \equiv (y_2/y_1)^{bt} \pmod{n}.$$

Dalej

$$(r_1 - r_2)t = \ell b + 1$$

dla pewnej liczby całkowitej  $\ell$ , wobec tego

$$v^{\ell b+1} \equiv (y_2/y_1)^{bt} \pmod{n},$$

lub równoważnie

$$v \equiv (y_2/y_1)^{bt}(v^{-1})^{\ell b} \pmod{n}.$$

Po podniesieniu obu stron kongruencji do potęgi  $b^{-1} \pmod{\phi(n)}$  otrzymamy

$$u^{-1} \equiv (y_2/y_1)^t(v^{-1})^\ell \pmod{n}.$$

Wreszcie, obliczając odwrotności modulo  $n$  obu stron, dochodzimy do następującej reprezentacji  $u$ :

$$u = (y_1/y_2)^t v^\ell \pmod{n}.$$

Wzór ten pozwala Oldze obliczyć  $u$  w czasie wielomianowym. ■

### Przykład 9.7

Jak w poprzednim przykładzie niech  $n = 223693$ ,  $b = 503$ ,  $u = 101576$  oraz  $v = 89888$ . Przypuśćmy, że Olga dowiedziała się, że

$$v^{401}103386^b \equiv v^{375}93725^b \pmod{n}.$$

Najpierw wykona ona następujące obliczenie:

$$\begin{aligned} t &= (r_1 - r_2)^{-1} \pmod{b} \\ &= (401 - 375)^{-1} \pmod{503} \\ &= 445, \end{aligned}$$

a następnie obliczy  $\ell$ :

$$\begin{aligned} \ell &= \frac{(r_1 - r_2)t - 1}{b} \\ &= \frac{(401 - 375)445 - 1}{503} \\ &= 23. \end{aligned}$$

Na koniec obliczy tajną wartość  $u$ :

$$\begin{aligned} u &= (y_1/y_2)^t v^\ell \pmod{n} \\ &= (103386/93725)^{445} 89888^{23} \pmod{223693} \\ &= 101576. \end{aligned}$$

W ten sposób skompromitowany został tajny wykładowik Alicji. ■

#### 9.4.1. Schematy identyfikacji oparte na tożsamości

Schemat identyfikacji Guillou–Quisquatera można przekształcić w schemat *oparty na tożsamości* (ang. *identity-based identification scheme*), charakteryzujący się przede wszystkim tym, że nie wymaga certyfikatów. Zamiast tego TA, znając ciąg ID Alicji, oblicza wartość  $u$  z wykorzystaniem publicznej funkcji skrótu  $h$  określonej na  $\mathbb{Z}_n$ . Przebieg tej operacji jest przedstawiony na rysunku 9.8.

- (1) TA ustala tożsamość Alicji i wystawia jej ciąg identyfikacyjny  $ID(\text{Alicja})$
  - (2) TA oblicza
- $$u = (h(\text{ID}(\text{Alicja}))^{-1})^a \bmod n$$
- i przekazuje Alicji wartość  $u$

**RYSUNEK 9.8. Przekazanie Alicji wartości  $u$**

Protokół identyfikacji działa teraz w sposób opisany na rysunku 9.9.

- (1) Alicja wybiera losową liczbę  $k$ , taką że  $0 \leq k \leq n - 1$ , i oblicza
- $$\gamma = k^b \bmod n$$
- (2) Alicja przekazuje Bolekowi  $\text{ID}(\text{Alicja})$  i wartość  $\gamma$
  - (3) Bolek oblicza
- $$v = h(\text{ID}(\text{Alicja}))$$
- (4) Bolek wybiera losową liczbę  $r$ , taką że  $0 \leq r \leq b - 1$ , i przekazuje ją Alicji
  - (5) Alicja oblicza
- $$y = k u^r \bmod n$$
- i przekazuje Bolekowi wynik  $y$
- (6) Bolek sprawdza, że
- $$\gamma \equiv v^r y^b \pmod{n}$$

**RYSUNEK 9.9. Schemat identyfikacji Guillou–Quisquatera oparty na tożsamości**

Również wartość  $v$  jest obliczana na podstawie ciągu ID Alicji za pomocą publicznej funkcji skrótu  $h$ . Aby zrealizować protokół identyfikacji, Alicja

musi znać wartość  $u$ , którą obliczyć może jedynie TA (przy założeniu, że system kryptograficzny RSA jest bezpieczny). Gdyby Olga chciała przedstawić się jako Alicja, poniosłaby fiasko właśnie z powodu nieznajomości  $u$ .

## 9.5. Przekształcenie identyfikacji w schemat podpisu

Istnieje standardowa metoda przekształcania schematu identyfikacji w schemat podpisu. Podstawowy pomysł polega na zastąpieniu czynnika weryfikującego (Bolka) przez publiczną funkcję skrótu  $h$ . W tak otrzymanym schemacie podpisu wiadomość nie jest skracana przed złożeniem podpisu, gdyż skracanie jest wkomponowane w algorytm podpisu.

Niech  $p$  będzie 512-bitową liczbą pierwszą, taką że problem logarytmu dyskretnego jest obliczeniowo nieroziągalny w  $\mathbb{Z}_p$ , i niech  $q$  będzie 160-bitowym dzielnicą pierwszym liczby  $p - 1$ . Niech  $\alpha \in \mathbb{Z}_p^*$  będzie pierwiastkiem stopnia  $q + 1$  modulo  $p$ . Dalej, niech  $h$  będzie funkcją skrótu o przeciwdziedzinie  $\mathbb{Z}_q$ . Przyjmijmy  $\mathcal{P} = \mathbb{Z}_p^*$ ,  $\mathcal{A} = \mathbb{Z}_p^* \times \mathbb{Z}_q$  oraz

$$\mathcal{K} = \{(p, q, \alpha, a, v) : v \equiv \alpha^{-a} \pmod{p}\}.$$

Wartości  $p$ ,  $q$ ,  $\alpha$  i  $v$  są publiczne, wartość  $a$  jest tajna.

Dla  $K = (p, q, \alpha, a, v)$  i (tajnej) liczby losowej  $k \in \mathbb{Z}_q^*$  definiujemy

$$sig_K(x, k) = (\gamma, y),$$

gdzie

$$\gamma = \alpha^k \pmod{p}$$

oraz

$$y = k + ah(x, \gamma) \pmod{q}.$$

Dla  $x, \gamma \in \mathbb{Z}_p^*$  i  $y \in \mathbb{Z}_q$  definiujemy

$$ver(x, \gamma, y) = \text{tak} \Leftrightarrow \gamma \equiv \alpha^v v^{h(x, \gamma)} \pmod{p}.$$

### RYSUNEK 9.10. Schemat podpisu Schnorra

Aby zilustrować takie podejście, przekształcimy schemat Schnorra w schemat podpisu (rysunek 9.10). W praktyce funkcją skrótu  $h$  byłby bezpieczny standard skrótu (SHS) z wynikiem zredukowanym modulo  $q$ . Zważywszy, że SHS produ-

kuje ciąg 160-bitowy, a  $q$  jest 160-bitową liczbą pierwszą, redukcja modulo  $q$  jest konieczna tylko wtedy, gdy skrót wiadomości uzyskany za pomocą SHS przekracza  $q$ ; wtedy wystarczy od wyniku odjąć  $q$ .

Przechodząc od schematu identyfikacji do schematu podpisu, zastąpiliśmy 40-bitowe pytanie 160-bitowym skrótem. Owe 40 bitów wystarczają do sformułowania pytania, gdyż ewentualny fałszerz musi odgadnąć pytanie, aby obliczyć akceptowalną odpowiedź. Jednakże w ramach schematu podpisu musimy mieć znacznie dłuższy skrót wiadomości, aby ustrzec się przed atakiem opartym na konfliktach funkcji skrótu.

Inne schematy identyfikacji mogą być przekształcane w schematy podpisu w podobny sposób.

---

## 9.6. Uwagi i bibliografia

Schemat identyfikacji Schnorra jest opisany w [SC91], schemat Okamoto przedstawiono w [OK93], natomiast schemat Guillou–Quisquatera można znaleźć w [GQ88]. Inny schemat, którego bezpieczeństwo można przy rozsądnych założeniach obliczeniowych udowodnić, podali Brickell i McCurley [BM92].

Wśród pozostałych popularnych schematów identyfikacji warto wymienić schemat Feigego–Fiata–Shamira [FFS88] (zobacz także [FS87]) oraz schemat permutacji jądra (ang. *permuted kernel scheme*) Shamira [SH90]. Bezpieczeństwo schematu Feigego–Fiata–Shamira zostało udowodnione za pomocą metody o wiedzy zerowej. (Rozdział 13 zawiera więcej informacji o dowodach o wiedzy zerowej).

Metodę budowania schematów podpisu ze schematów identyfikacji opracowali Fiat i Shamir [FS87]. Opisali oni również wersję swojego schematu identyfikacji opartą na tożsamości.

Burmester, Desmedt i Beth [BDB92] oraz de Waleffe i Quisquater [DWQ93] opublikowali przegląd schematów identyfikacji.

---

## Ćwiczenia

- 9.1. Rozważmy pewien hipotetyczny schemat identyfikacji. Alicja ma tajny klucz  $n = pq$ , gdzie  $p$  i  $q$  są liczbami pierwszymi oraz  $p \equiv q \equiv 3 \pmod{4}$ . Wartości  $n$  i  $\text{ID}(\text{Alicja})$  są, jak zwykle, podpisane przez TA i zapisane w certyfikacie Alicji. Gdy Alicja chce się przedstawić, na przykład Bolkowi, ten przekazuje jej losową resztę kwadratową  $x$  modulo  $n$ . Wtedy Alicja oblicza pierwiastek kwadratowy  $y$  z liczby  $x$  i przesyła wynik Bolkowi. Bolek sprawdza, czy  $y^2 \equiv x \pmod{n}$ . Wyjaśnij, dlaczego ten schemat nie jest bezpieczny.

- 9.2. Założmy, że Alicja używa schematu Schnorra z wartościami  $q = 1201$ ,  $p = 122503$ ,  $t = 10$  oraz  $\alpha = 11538$ .
- Sprawdź, że  $\alpha$  ma w  $\mathbb{Z}_p^*$  rząd  $q$ .
  - Oblicz  $v$ , jeśli tajnym wykładownikiem Alicji jest  $a = 357$ .
  - Oblicz  $\gamma$ , jeśli  $k = 868$ .
  - Niech pytaniem Bolka będzie  $r = 501$ . Oblicz odpowiedź Alicji  $y$ .
  - Wykonaj za Bolka obliczenia sprawdzające  $y$ .

- 9.3. Założmy, że Alicja używa schematu Schnorra z wartościami  $p$ ,  $q$ ,  $t$  i  $\alpha$  jak w ćwiczeniu 9.2. Przypuśćmy, że  $v = 51131$ , a Olga dowiedziała się, że

$$\alpha^3 v^{148} \equiv \alpha^{151} v^{1077} \pmod{p}.$$

Wyjaśnij, jak Olga może obliczyć tajny wykładownik Alicji  $\alpha$ .

- 9.4. Założmy, że Alicja używa schematu Okamoto z wartościami  $q = 1201$ ,  $p = 122503$ ,  $t = 10$ ,  $\alpha_1 = 60497$  oraz  $\alpha_2 = 17163$ .
- Oblicz  $v$ , jeśli tajnymi wykładownikami Alicji są  $a_1 = 432$  oraz  $a_2 = 423$ .
  - Oblicz  $\gamma$ , jeśli  $k_1 = 389$  i  $k_2 = 191$ .
  - Bolek wysłał pytanie  $r = 21$ . Oblicz wartości  $y_1$  i  $y_2$ , składające się na odpowiedź Alicji.
  - Wykonaj za Bolka obliczenia sprawdzające  $y_1$  i  $y_2$ .

- 9.5. Założmy, że Alicja używa schematu Okamoto z wartościami  $p$ ,  $q$ ,  $t$ ,  $\alpha_1$  i  $\alpha_2$  jak w ćwiczeniu 9.4. Założymy też, że  $v = 119504$ .

- (a) Sprawdź, że

$$\alpha_1^{70} \alpha_2^{1033} v^{877} \equiv \alpha_1^{248} \alpha_2^{883} v^{992} \pmod{p}.$$

- (b) Wykorzystaj tę informację do obliczenia  $b_1$  i  $b_2$ , takich że

$$\alpha_1^{-b_1} \alpha_2^{-b_2} \equiv v \pmod{p}.$$

- (c) Założymy teraz, że Alicja ujawnia wartości  $a_1 = 484$  i  $a_2 = 935$ . Wyjaśnij, jak Alicja razem z Olgą mogą obliczyć  $\log_{\alpha_1} \alpha_2$ .

- 9.6. Założmy, że Alicja używa schematu Guillou–Quisquatera z wartościami  $p = 503$ ,  $q = 379$  oraz  $b = 509$ .
- Oblicz  $v$ , jeśli tajna wartość  $u$  Alicji jest równa 155863.
  - Oblicz  $\gamma$ , jeśli  $k = 123845$ .
  - Oblicz wartość  $y$ , którą Alicja wyśle w odpowiedzi na pytanie Bolka  $r = 487$ .
  - Wykonaj za Bolka obliczenia sprawdzające  $y$ .

- 9.7. Założmy, że Alicja używa schematu Guillou–Quisquatera z wartościami  $n = 199543$ ,  $b = 523$  i  $v = 146152$ . Przypuśćmy, że Oldze udało się ustalić, że

$$v^{456} 101360^b \equiv v^{257} 36056^b \pmod{n}.$$

Wyjaśnij, jak Olga może obliczyć  $u$ .

# 10

---

## Kody uwierzytelniania

---

### 10.1. Wprowadzenie

Poświęciliśmy wiele miejsca systemom kryptograficznym służącym do utajniania wiadomości. Kod uwierzytelnienia pozwala zapewnić *nienaruszalność* (ang. *integrity*) wiadomości, a więc utwierdzić odbiorcę w przekonaniu, że wiadomość nie uległa przeróbce, a utworzyła ją rzeczywiście osoba podająca się za nadawcę. Chcemy uzyskać taką możliwość uwierzytelniania nawet w obecności aktywnego przeciwnika, Oskara, obserwującego wiadomości przesyłane na całym komunikacyjnym i wprowadzającego doń własne wiadomości. Zrealizujemy ten cel w ramach systemu z kluczem prywatnym, w którym przed przekazaniem pierwszej wiadomości Alicja i Bolek wspólnie korzystają z tajnego klucza  $K$ .

W tym rozdziale zajmiemy się kodami zapewniającymi uwierzytelnienie, ale nie tajność. W tego typu kodzie klucz służy do obliczenia etykiety uwierzytelniającej (ang. *authentication tag*), która pozwoli Bolkowi sprawdzić autentyczność otrzymanej wiadomości. Kod uwierzytelnienia (ang. *authentication code*) pozwala również sprawdzić, czy wiadomość dotarła nienaruszona. Etykieta uwierzytelniająca jest zapisywana wraz z danymi; klucz użyty do jej wygenerowania i sprawdzenia wiarygodności osoby uwierzytelniającej jest przechowywany oddziennie, w „bezpiecznej” strefie.

Należy tu także podkreślić, że pod wieloma względami kod uwierzytelnienia jest podobny do schematu podpisu lub kodu uwierzytelniania wiadomości (ang. *message authentication code*, MAC). Są jednak pewne różnice. Kod uwierzytelniania zapewnia bezwarunkowe bezpieczeństwo, podczas gdy w schematach podpisu i w MAC rozuwa się bezpieczeństwo obliczeniowe. Ponadto, przy stosowaniu kodu uwierzytelniania (lub MAC) tylko planowany nadawca może sprawdzić wiadomość, choć każdy może zweryfikować podpis za pomocą publicznego algorytmu weryfikacji.

Podamy teraz formalną definicję pojęć, którymi będziemy się posługiwali przy badaniu kodów uwierzytelnienia.

**DEFINICJA 10.1**

Kodem uwierzytelniania nazywamy czwórkę  $(\mathcal{S}, \mathcal{A}, \mathcal{K}, \mathcal{E})$ , gdy spełnione są następujące warunki:

- (1)  $\mathcal{S}$  jest skończonym zbiorem możliwych stanów źródłowych.
- (2)  $\mathcal{A}$  jest skończonym zbiorem możliwych etykiet uwierzytelniających.
- (3)  $\mathcal{K}$ , przestrzeń kluczy, jest skończonym zbiorem możliwych kluczy.
- (4) Dla każdego  $K \in \mathcal{K}$  dana jest reguła uwierzytelniania  $e_K : \mathcal{S} \rightarrow \mathcal{A}$ .

Zbiór  $\mathcal{M} = \mathcal{S} \times \mathcal{A}$  nazywamy zbiorem wiadomości.

**UWAGA** Zauważmy tu analogię między stanem źródłowym a tekstem jawnym. Wiadomość składa się z tekstu jawnego i dołączonej etykiety uwierzytelniającej; moglibyśmy ją nazwać dokładniej *podpisaną wiadomością*. Ponadto, reguła uwierzytelniania nie musi być funkcją różnowartościową. ■

Gdy Alicja chce przekazać Bolkowi (podpisaną) wiadomość, oboje przystępują do realizacji opowiedniego protokołu. Po pierwsze, wspólnie wybierają losowo klucz  $K \in \mathcal{K}$ ; czynią to potajemnie, jak zwykle w systemach kryptograficznych z kluczem prywatnym. Gdy później Alicja zechce przesłać Bolkowi stan źródłowy  $s \in \mathcal{S}$  za pośrednictwem niepewnego kanału komunikacyjnego, obliczy  $a = e_K(s)$  i wyśle Bolkowi parę  $(s, a)$ . Po jej otrzymaniu Bolek liczy  $a' = e_K(s)$ ; zaakceptuje wiadomość jako autentyczną, jeśli  $a' = a$ , w przeciwnym przypadku odrzuci ją.

Zbadamy dwa różne rodzaje możliwego ataku Oskara. W obu rozważanych przypadkach występuje on jako intruz-w-środku. Oto krótki opis obu wariantów:

**Podszywanie się pod nadawcę.** Oskar wprowadza do kanału komunikacyjnego wiadomość  $(s, a)$ , licząc na to, że Bolek uzna ją za autentyczną. Opisuje to rysunek 10.1.



RYSUNEK 10.1. Oskar podszywa się

**Podstawnienie.** Oskar widzi w kanale komunikacyjnym wiadomość  $(s, a)$  i podmienia ją wiadomością  $(s', a')$ , gdzie  $s' \neq s$ . Znów liczy na to, że Bolek przyjmie ją za wiadomość autentyczną. W ten sposób podsusza Bolkowi fałszywy stan źródłowy. Ilustruje to rysunek 10.2.



RYSUNEK 10.2. Podstawienie w wykonaniu Oskara

Z każdym z tych ataków wiąże się *prawdopodobieństwo oszustwa*, czyli tego, że Bolek da się oszukać, jeśli Oskar będzie realizował optymalną strategię. Oznaczmy je symbolami  $Pd_0$  (dla podszywania się) i  $Pd_1$  (dla podstawiania). Wyznaczenie tych prawdopodobieństw wymaga ustalenia rozkładów prawdopodobieństwa na zbiorach  $\mathcal{S}$  i  $\mathcal{K}$ ; oznaczmy te rozkłady, odpowiednio,  $p_{\mathcal{S}}$  i  $p_{\mathcal{K}}$ . Zakładamy, że Oskar zna kod uwierzytelniania oraz oba rozkłady prawdopodobieństwa. Jedyną informacją, którą dysponują tylko Alicja i Bolek, jest wartość klucza  $K$ . Kontekst jest więc podobny do tego, który rozważaliśmy, badając bezwarunkowe bezpieczeństwo systemu kryptograficznego z kluczem prywatnym.

## 10.2. Obliczanie prawdopodobieństw oszustwa

Przyjrzyjmy się obliczeniu prawdopodobieństwa oszustwa. Zaczniemy od prostego przykładu kodu uwierzytelniania.

*Przykład 10.1*

Załóżmy, że

$$\mathcal{S} = \mathcal{A} = \mathbb{Z}_3$$

oraz

$$\mathcal{K} = \mathbb{Z}_3 \times \mathbb{Z}_3.$$

Dla każdej pary  $(i, j) \in \mathcal{K}$  i dowolnego  $s \in \mathcal{S}$  niech

$$e_{ij}(s) = is + j \bmod 3.$$

Warto tu wprowadzić do rozważań *macierz uwierzytelniania*, zawierającą wszystkie wartości  $e_{ij}(s)$ . Dla każdego klucza  $K \in \mathcal{K}$  oraz  $s \in \mathcal{S}$  umieścmy etykietę uwierzytelniającą  $e_K(s)$  w wierszu o numerze  $K$  i w kolumnie o numerze  $s$  macierzy  $M$  wymiaru  $|\mathcal{K}| \times |\mathcal{S}|$ . Na rysunku 10.3 przedstawiamy macierz  $M$ .

Załóżmy, że klucz jest wybierany losowo; wówczas  $p_{\mathcal{K}}(K) = 1/9$  dla każdego  $K \in \mathcal{K}$ . Nie określamy tu rozkładu prawdopodobieństwa  $p_{\mathcal{S}}$ , gdyż nie jest on w tym przykładzie istotny.

Rozważmy najpierw atak polegający na podszywaniu się pod innego użytkownika. Oskar wybiera stan źródłowy  $s$  i stara się odgadnąć „poprawną” etykietę uwierzytelniającą. Niech  $K_0$  oznacza rzeczywisty, nieznany Oskarowi aktualny

| Klucz | 0 | 1 | 2 |
|-------|---|---|---|
| (0,0) | 0 | 0 | 0 |
| (0,1) | 1 | 1 | 1 |
| (0,2) | 2 | 2 | 2 |
| (1,0) | 0 | 1 | 2 |
| (1,1) | 1 | 2 | 0 |
| (1,2) | 2 | 0 | 1 |
| (2,0) | 0 | 2 | 1 |
| (2,1) | 1 | 0 | 2 |
| (2,2) | 2 | 1 | 0 |

RYSUNEK 10.3. Macierz uwierzytelniania

klucz. Oskarowi uda się oszukać Bolka, jeśli odgadnie etykietę  $a_0 = e_{K_0}(s)$ . Łatwo jednak sprawdzić, że dla dowolnej pary  $s \in \mathcal{S}$  i  $a \in \mathcal{A}$  istnieją dokładnie trzy (spośród dziewięciu) reguły uwierzytelniania  $K \in \mathcal{K}$ , takie że  $e_K(s) = a$ . (Innymi słowy, każdy symbol występuje trzykrotnie w każdej kolumnie macierzy uwierzytelniania). Wynika stąd, że  $Pd_0 = 1/3$ .

Analiza ataku przez podstawienie jest nieco bardziej złożona. Przypuśćmy, dla ustalenia uwagi, że Oskar wypatrzył w kanale komunikacyjnym wiadomość  $(0, 0)$ . Kryje się w tym jakaś informacja o kluczu. Teraz Oskar wie, że

$$K_0 \in \{(0, 0), (1, 0), (2, 0)\}.$$

Przyjmijmy, że Oskar zastępuje wiadomość  $(0, 0)$  parą  $(1, 1)$ . To oszustwo się powiedzie wtedy i tylko wtedy, gdy  $K_0 = (1, 0)$ . Jeśli wiemy, że rzeczywisty klucz  $K_0$  jest w zbiorze  $\{(0, 0), (1, 0), (2, 0)\}$ , to prawdopodobieństwo trafienia jest znów równe  $1/3$ .

Podobne rozumowanie odnosi się do dowolnego podstawienia, które może wykonać Oskar. Ogólniej, jeśli Oskar widzi wiadomość  $(s, a)$  i zastępuje ją wiadomością  $(s', a')$ , gdzie  $s \neq s'$ , to prawdopodobieństwo oszukania Bolka jest równe  $1/3$ . Istotnie, zauważenie wiadomości  $(s, a)$  ogranicza poszukiwanego klucza do trzech możliwości. Dalej, dla każdego wyboru pary  $(s', a')$ , jest tylko jeden klucz (z trzech możliwych), przy którym  $a'$  jest etykietą uwierzytelniającą dla  $s'$ .  $\square$

Zastanówmy się teraz, jak obliczać prawdopodobieństwa oszustwa w przypadku ogólnym. Rozważmy najpierw  $Pd_0$ . Jak poprzednio, niech  $K_0$  oznacza klucz wybrany przez Alicję i Bolka. Dla  $s \in \mathcal{S}$  i  $a \in \mathcal{A}$  niech  $po(s, a)$  oznacza prawdopodobieństwo tego, że Bolek uzna wiadomość  $(s, a)$  za autentyczną. Nietrudno zauważyć, że

$$\begin{aligned} po(s, a) &= P(a = e_{K_0}(s)) \\ &= \sum_{\{K \in \mathcal{K}: e_K(s)=a\}} p_{\mathcal{K}}(K). \end{aligned}$$

Oznacza to, że w celu obliczenia wartości  $po(s, a)$  należy wybrać te wiersze macierzy uwierzytelniania, w których w kolumnie  $s$  występuje  $a$  i dodać prawdopodobieństwa odpowiadających im kluczy.

Gdy Oskar zechce zmaksymalizować szanse powodzenia oszustwa, wybierze taką parę  $(s, a)$ , dla której  $po(s, a)$  ma największą wartość. Stąd

$$Pd_0 = \max\{po(s, a) : s \in \mathcal{S}, a \in \mathcal{A}\}. \quad (10.1)$$

Odnotujmy, że  $Pd_0$  nie zależy od rozkładu prawdopodobieństwa  $p_{\mathcal{S}}$ .

Obliczenie  $Pd_1$  jest nieco bardziej skomplikowane i zależy już od rozkładu  $p_{\mathcal{S}}$ . Rozważmy najpierw inny problem. Przypuśćmy, że Oskar widzi w kanale komunikacyjnym wiadomość  $(s, a)$  i zamienia ją na parę  $(s', a')$ , przy czym  $s' \neq s$ . Dla  $s, s' \in \mathcal{S}$ , takich że  $s' \neq s$ , oraz dla  $a, a' \in \mathcal{A}$  niech  $po(s', a'; s, a)$  oznacza prawdopodobieństwo tego, że podstawienie  $(s', a')$  za  $(s, a)$  wprowadzi Bolka w błąd. Możemy wówczas przeprowadzić następujące obliczenie:

$$\begin{aligned} po(s', a'; s, a) &= P(a' = e_{K_0}(s') | a = e_{K_0}(s)) \\ &= \frac{P(a' = e_{K_0}(s') \wedge a = e_{K_0}(s))}{P(a = e_{K_0}(s))} \\ &= \frac{\sum_{\{K \in \mathcal{K}: e_K(s)=a, e_K(s')=a'\}} p_{\mathcal{K}}(K)}{\sum_{\{K \in \mathcal{K}: e_K(s)=a\}} p_{\mathcal{K}}(K)} \\ &= \frac{\sum_{\{K \in \mathcal{K}: e_K(s)=a, e_K(s')=a'\}} p_{\mathcal{K}}(K)}{po(s, a)} \end{aligned}$$

Licznik w tym ułamku jest sumą prawdopodobieństw kluczy odpowiadających tym wierszom macierzy uwierzytelniania, w których w kolumnie  $s$  występuje  $a$  i jednocześnie w kolumnie  $s'$  występuje  $a'$ .

Oskar, chcąc zmaksymalizować szansę oszukania Bolka, obliczy

$$p_{s,a} = \max\{po(s', a'; s, a) : s' \in \mathcal{S}, s \neq s', a' \in \mathcal{A}\}.$$

Wartość  $p_{s,a}$  określa prawdopodobieństwo oszukania Bolka za pomocą podstawienia, gdy obserwowaną przez Oskara wiadomością jest para  $(s, a)$ .

Jak więc teraz obliczyć prawdopodobieństwo oszustwa  $Pd_1$ ? Rzeczą jasna musimy obliczyć średnią ważoną wartości  $p_{s,a}$  względem prawdopodobieństw  $p_{\mathcal{M}}(s, a)$  wychwycenia w kanale wiadomości  $(s, a)$ . Tak więc,

$$Pd_1 = \sum_{(s,a) \in \mathcal{M}} p_{\mathcal{M}}(s, a) p_{s,a}. \quad (10.2)$$

Rozkład prawdopodobieństwa  $p_{\mathcal{M}}$  określony jest następująco:

$$\begin{aligned} p_{\mathcal{M}}(s, a) &= p_{\mathcal{S}}(s) \times p_{\mathcal{K}}(a|s) \\ &= p_{\mathcal{S}}(s) \times \sum_{\{K \in \mathcal{K}: e_K(s)=a\}} p_{\mathcal{K}}(K) \\ &= p_{\mathcal{S}}(s) \times po(s, a). \end{aligned}$$

W przykładzie 10.1

$$po(s, a) = 1/3$$

dla wszystkich  $s, a$ , zatem  $Pd_0 = 1/3$ . Można również sprawdzić, że

$$po(s', a'; s, a) = 1/3$$

dla wszystkich  $s, s', a, a', s \neq s'$ . Dlatego  $Pd_1 = 1/3$  dla dowolnego rozkładu prawdopodobieństwa  $p_S$  (na ogólny jednak  $Pd_1$  zależy od  $p_S$ ).

Popatrzmy, jak wyglądają obliczenia  $Pd_0$  i  $Pd_1$  w mniej typowym przypadku.

| Klucz | 1 | 2 | 3 | 4 |
|-------|---|---|---|---|
| 1     | 1 | 1 | 1 | 2 |
| 2     | 2 | 2 | 1 | 2 |
| 3     | 1 | 2 | 2 | 1 |

RYSUNEK 10.4. Macierz uwierzytelniania

### Przykład 10.2

Weźmy pod uwagę macierz uwierzytelniania z rysunku 10.4. Założymy, że rozkłady prawdopodobieństw na  $\mathcal{S}$  i  $\mathcal{K}$  są określone następująco:

$$p_{\mathcal{S}}(i) = 1/4$$

dla  $1 \leq i \leq 4$  oraz

$$p_{\mathcal{K}}(1) = 1/2,$$

$$p_{\mathcal{K}}(2) = p_{\mathcal{K}}(3) = 1/4.$$

Wówczas

$$po(1, 1) = 3/4$$

$$po(1, 2) = 1/4$$

$$po(2, 1) = 1/2$$

$$po(2, 2) = 1/2$$

$$po(3, 1) = 3/4$$

$$po(3, 2) = 1/4$$

$$po(4, 1) = 1/4$$

$$po(4, 2) = 3/4.$$

W rezultacie  $Pd_0 = 3/4$ . Optymalna strategia Oskara polega zatem na umieszczeniu w kanale jednej z trzech wiadomości: (1, 1), (3, 1) lub (4, 2).

Przejdźmy teraz do obliczenia  $Pd_1$ . Zaczniemy od przedstawienia wartości  $po(s', a'; s, a)$  w postaci macierzy. Liczba w wierszu  $(s, a)$  i kolumnie  $(s', a')$  oznacza wartość  $po(s', a'; s, a)$ .

|       | (1,1) | (1,2) | (2,1) | (2,2) | (3,1) | (3,2) | (4,1) | (4,2) |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| (1,1) |       |       | 2/3   | 1/3   | 2/3   | 1/3   | 1/3   | 2/3   |
| (1,2) |       |       | 0     | 1     | 1     | 0     | 1     | 0     |
| (2,1) | 1     | 0     |       |       | 0     | 1     | 0     | 1     |
| (2,2) | 1/2   | 1/2   |       |       | 1/2   | 1/2   | 1/2   | 1/2   |
| (3,1) | 2/3   | 1/3   | 2/3   | 1/3   |       |       | 0     | 1     |
| (3,2) | 1     | 0     | 0     | 1     |       |       | 1     | 0     |
| (4,1) | 1     | 0     | 0     | 1     | 0     | 1     |       |       |
| (4,2) | 2/3   | 1/3   | 2/3   | 1/3   | 1     | 0     |       |       |

Mamy zatem  $p_{1,1} = 2/3$ ,  $p_{2,2} = 1/2$  oraz  $p_{s,a} = 1$  dla wszystkich pozostałych wartości  $s, a$ . Obliczamy teraz bez trudu  $Pd_1 = 7/8$ . Optymalną strategię podstawieniową Oskara można opisać w następujący sposób:

$$\begin{aligned}(1,1) &\rightarrow (2,1) \\(1,2) &\rightarrow (2,2) \\(2,1) &\rightarrow (1,1) \\(2,2) &\rightarrow (1,1) \\(3,1) &\rightarrow (4,2) \\(3,2) &\rightarrow (1,1) \\(4,1) &\rightarrow (1,1) \\(4,2) &\rightarrow (3,1)\end{aligned}$$

Przy tej strategii rzeczywiście  $Pd_1 = 7/8$ . □

Obliczanie  $Pd_1$  w przykładzie 10.2 jest proste, lecz żmudne. Możemy je skrócić, jeśli zauważymy, że przy obliczaniu  $p_{s,a}$  dzielimy przez  $po(s,a)$ , by potem mnożyć przez  $po(s,a)$ , gdy obliczamy  $Pd_1$ . Oczywiście oba te działania znoszą się wzajemnie. Niech

$$q_{s,a} = \max \left\{ \sum_{\{K \in \mathcal{K}: e_K(s)=a, e_K(s')=a'\}} p_K(K) : s' \in \mathcal{S}, s' \neq s, a' \in \mathcal{A} \right\}$$

dla dowolnych  $s, a$ . Mamy wówczas następującą zwięzłą postać  $Pd_1$ :

$$Pd_1 = \sum_{(s,a) \in \mathcal{M}} p_{\mathcal{S}}(s) q_{s,a}. \quad (10.3)$$

### 10.3. Ograniczenia kombinatoryczne

Stwierdziliśmy już, że miarą bezpieczeństwa kodu uwierzytelniania są prawdopodobieństwa oszustwa. Chcielibyśmy zatem projektować takie kody, dla których prawdopodobieństwa te są jak najmniejsze. Ważne są jednak także inne wzgłydy. Rozważmy kilka własności, które powinien mieć dobry kod uwierzytelniania.

- (1) Prawdopodobieństwa  $Pd_0$  i  $Pd_1$  muszą być dostatecznie małe, by zapewnić oczekiwany poziom bezpieczeństwa.
- (2) Liczba stanów źródłowych powinna być dostatecznie duża, byśmy mogli przy przekazywaniu informacji dodawać jedną etykietę uwierzytelniającą do jednego stanu źródłowego.
- (3) Wielkość przestrzeni kluczy powinna być zminimalizowana, gdyż wartość klucza musi być przesyłana bezpiecznym kanałem. Zauważmy, że klucz zmienia się z każdą wiadomością, podobnie jak w przypadku szyfru z kluczem jednorazowym.

W tym podrozdziale ustalimy dolne ograniczenia prawdopodobieństw oszustwa, wskazując na ich zależność od pozostałych parametrów kodu. Przypomnijmy, że zdefiniowaliśmy kod uwierzytelniania jako czwórkę  $(\mathcal{S}, \mathcal{A}, \mathcal{K}, \mathcal{E})$ . W całym tym podrozdziale przyjmiemy oznaczenie  $\ell = |\mathcal{A}|$ .

Niech  $s \in \mathcal{S}$  będzie ustalonym stanem źródłowym. Obliczamy:

$$\begin{aligned} \sum_{a \in \mathcal{A}} po(s, a) &= \sum_{a \in \mathcal{A}} \sum_{\{K \in \mathcal{K}: e_K(s) = a\}} p_K(K) \\ &= \sum_{K \in \mathcal{K}} p_K(K) \\ &= 1. \end{aligned}$$

Stąd wynika, że dla każdego stanu  $s \in \mathcal{S}$  istnieje etykieta uwierzytelniająca  $a(s)$ , taka że

$$po(s, a(s)) \geq \frac{1}{\ell}.$$

Łatwo otrzymujemy następujące twierdzenie.

#### TWIERDZENIE 10.1

Niech  $(\mathcal{S}, \mathcal{A}, \mathcal{K}, \mathcal{E})$  będzie kodem uwierzytelniania. Wówczas  $Pd_0 \geq 1/\ell$ , gdzie  $\ell = |\mathcal{A}|$ . Ponadto,  $Pd_0 1/\ell$  wtedy i tylko wtedy, gdy

$$\sum_{\{K \in \mathcal{K}: e_K(s) = a\}} p_K(K) = \frac{1}{\ell}. \quad (10.4)$$

dla każdego  $s \in \mathcal{S}$  i  $a \in \mathcal{A}$ .

Zajmijmy się teraz podstawieniem. Ustalmy wartości  $s$ ,  $a$  i  $s'$ , przy czym  $s \neq s'$ . Mamy wtedy:

$$\begin{aligned} \sum_{a' \in \mathcal{A}} po(s', a'; s, a) &= \sum_{a' \in \mathcal{A}} \frac{\sum_{\{K \in \mathcal{K}: e_K(s)=a, e_K(s')=a'\}} p_{\mathcal{K}}(K)}{\sum_{\{K \in \mathcal{K}: e_K(s)=a\}} p_{\mathcal{K}}(K)} \\ &= \frac{\sum_{\{K \in \mathcal{K}: e_K(s)=a\}} p_{\mathcal{K}}(K)}{\sum_{\{K \in \mathcal{K}: e_K(s)=a\}} p_{\mathcal{K}}(K)} \\ &= 1. \end{aligned}$$

Istnieje zatem etykieta uwierzytelniająca  $a'(s', s, a)$ , taka że

$$po(s', a'(s', s, a); s, a) \geq \frac{1}{\ell}.$$

Z ostatniego twierdzenia otrzymujemy następujące wnioski.

### TWIERDZENIE 10.2

Niech  $(\mathcal{S}, \mathcal{A}, \mathcal{K}, \mathcal{E})$  będzie kodem uwierzytelniania. Wówczas  $Pd_1 \geq 1/\ell$ , gdzie  $\ell = |\mathcal{A}|$ . Ponadto  $Pd_1 = 1/\ell$  wtedy i tylko wtedy, gdy

$$\frac{\sum_{\{K \in \mathcal{K}: e_K(s)=a, e_K(s')=a'\}} p_{\mathcal{K}}(K)}{\sum_{\{K \in \mathcal{K}: e_K(s)=a\}} p_{\mathcal{K}}(K)} = \frac{1}{\ell} \quad (10.5)$$

dla dowolnych  $s, s' \in \mathcal{S}$ ,  $s' \neq s$ ,  $a, a' \in \mathcal{A}$ .

### DOWÓD

Mamy

$$\begin{aligned} Pd_1 &= \sum_{(s,a) \in \mathcal{M}} p_{\mathcal{M}}(s, a) p_{s,a} \\ &\geq \sum_{(s,a) \in \mathcal{M}} \frac{p_{\mathcal{M}}(s, a)}{\ell} \\ &= \frac{1}{\ell}. \end{aligned}$$

Równość zachodzi wtedy i tylko wtedy, gdy  $p_{s,a} = 1/\ell$  dla każdej pary  $(s, a)$ , a ten warunek jest z kolei równoważny temu, że  $po(s, a) = 1/\ell$  dla każdej pary  $(s, a)$ . ■

Łącząc twierdzenia 10.1 i 10.2, otrzymujemy:

**TWIERDZENIE 10.3**

Niech  $(\mathcal{S}, \mathcal{A}, \mathcal{K}, \mathcal{E})$  będzie kodem uwierzytelniania i niech  $\ell = |\mathcal{A}|$ . Wtedy  $Pd_0 = Pd_1 = 1/\ell$  wtedy i tylko wtedy, gdy

$$\sum_{\{K \in \mathcal{K}: e_K(s) = a, e_K(s') = a'\}} p_K(K) = \frac{1}{\ell^2} \quad (10.6)$$

dla dowolnych  $s, s' \in \mathcal{S}$ ,  $s' \neq s$ ,  $a, a' \in \mathcal{A}$ .

**DOWÓD** Równanie (10.6) wynika z równań (10.4) i (10.5) – i na odwrót: równania (10.4) i (10.5) wynikają z równania (10.6). ■

Jeśli klucze są jednakowoprawdopodobne, otrzymujemy następujący wniosek:

**WNIOSEK 10.4**

Niech  $(\mathcal{S}, \mathcal{A}, \mathcal{K}, \mathcal{E})$  będzie kodem uwierzytelniania i niech  $\ell = |\mathcal{A}|$ . Jeśli klucze są wybierane z jednakowym prawdopodobieństwem, to  $Pd_0 = Pd_1 = 1/\ell$  wtedy i tylko wtedy, gdy

$$|\{K \in \mathcal{K}: e_K(s) = a, e_K(s') = a'\}| = \frac{|\mathcal{K}|}{\ell^2} \quad (10.7)$$

dla dowolnych  $s, s' \in \mathcal{S}$ ,  $s' \neq s$ ,  $a, a' \in \mathcal{A}$ .

**10.3.1. Macierze ortogonalne**

Zajmiemy się tu związkami między kodami uwierzytelniania i pewnymi strukturami kombinatorycznymi zwanyymi macierzami ortogonalnymi. Zaczniemy od definicji.

**DEFINICJA 10.2**

Macierzą ortogonalną  $MO(n, k, \lambda)$  nazywamy macierz  $\lambda n^2 \times k$  nad zbiorem  $n$ -elementowym, taką że w dowolnych dwóch jej kolumnach każda z możliwych  $n^2$  par elementów występuje w dokładnie  $\lambda$  wierszach.

W teorii konfiguracji kombinatorycznych macierze ortogonalne są strukturami dobrze zbadanymi; są one równoważne innym znany strukturom, takim jak konfiguracje przekątniowe, wzajemnie ortogonalne kwadraty łacińskie i sieci.

Na rysunku 10.5 przedstawiamy  $MO(3, 3, 1)$  otrzymaną z macierzy uwierzytelniania z rysunku 10.3. Dowolna macierz ortogonalna  $MO(n, k, \lambda)$  może posłużyć do konstrukcji kodu uwierzytelniania z  $Pd_0 = Pd_1 = 1/n$ , co stwierdza następujące twierdzenie.

**TWIERDZENIE 10.5**

Niech  $MO(n, k, \lambda)$  będzie macierzą ortogonalną. Wówczas istnieje kod uwierzytelniania  $(\mathcal{S}, \mathcal{A}, \mathcal{K}, \mathcal{E})$ , taki że  $|\mathcal{S}| = k$ ,  $|\mathcal{A}| = n$ ,  $|\mathcal{K}| = \lambda n^2$  oraz  $Pd_0 = Pd_1 = 1/n$ .

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 2 & 2 & 2 \\ 0 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 1 \\ 0 & 2 & 1 \\ 1 & 0 & 2 \\ 2 & 1 & 0 \end{pmatrix}$$

RYSUNEK 10.5. MO(3, 3, 1)

**DOWÓD** Wystarczy użyć każdego wiersza macierzy ortogonalnej jako reguły uwierzytelniania z jednakowym prawdopodobieństwem  $1/(\lambda n^2)$ . Mamy wtedy następujące odpowiedniości:

| Macierz ortogonalna | Kod uwierzytelniania       |
|---------------------|----------------------------|
| wiersz              | reguła uwierzytelniania    |
| kolumna             | stan źródłowy              |
| symbol              | etykieta uwierzytelniająca |

Możemy teraz odwołać się do wniosku 10.4, gdyż spełnione jest równanie (10.7). W rezultacie otrzymujemy kod o żądanach własnościach. ■

### 10.3.2. Macierze ortogonalne – konstrukcje i ograniczenia

Założymy, że budujemy kod uwierzytelniania z macierzy  $MO(n, k, \lambda)$ . Parametr  $n$  określa liczbę etykiet uwierzytelniających (a więc bezpieczeństwo systemu), podczas gdy  $k$  wyznacza liczbę stanów źródłowych, które kod może przetworzyć. Parametr  $\lambda$  odnosi się jedynie do liczby kluczowej, równej  $\lambda n^2$ . Oczywiście, najbardziej pożądana jest przypadek  $\lambda = 1$ , ale jak zobaczymy, niekiedy konieczne staje się użycie macierzy ortogonalnych z większymi wartościami  $\lambda$ .

Przypuśćmy, że chcemy zbudować kod uwierzytelniania z określonym zbiorem stanów źródłowych  $\mathcal{S}$  i ustalonym poziomem bezpieczeństwa  $\epsilon$  (czyli tak, aby  $Pd_0 \leq \epsilon$  oraz  $Pd_1 \leq \epsilon$ ). Odpowiednia macierz ortogonalna musi zatem spełniać następujące warunki:

- (1)  $n \geq 1/\epsilon$ ;
- (2)  $k \geq |\mathcal{S}|$  (zauważmy, że zawsze możemy usunąć z macierzy ortogonalnej jedną lub więcej kolumn, nie naruszając jej ortogonalności; dlatego nie musimy wymagać, by  $k = |\mathcal{S}|$ );
- (3)  $\lambda$  ma najmniejszą możliwą wartość, która pozwala spełnić poprzednie dwa warunki.

Zajmijmy się najpierw macierzami ortogonalnymi z  $\lambda = 1$ . Przy ustalonej wartości  $n$  chcemy zmaksymalizować liczbę kolumn. Oto warunek konieczny istnienia odpowiedniej macierzy.

### TWIERDZENIE 10.6

Jeśli istnieje macierz  $\text{MO}(n, k, 1)$ , to  $k \leq n + 1$ .

**DOWÓD** Niech  $A$  będzie macierzą typu  $\text{MO}(n, k, 1)$  nad zbiorem symboli  $X = \{0, 1, \dots, n - 1\}$ . Jeśli dokonamy permutacji symboli w dowolnej kolumnie macierzy  $A$ , otrzymamy ponownie macierz typu  $\text{MO}(n, k, 1)$ . Tak więc, stosując w razie potrzeby pewien ciąg permutacji tego rodzaju, możemy przyjąć bez utraty ogólności, że pierwszy wiersz macierzy  $A$  składa się z samych zer:  $(00\dots 0)$ .

Wykażemy teraz, że każdy symbol musi wystąpić dokładnie  $n$  razy w każdej kolumnie  $A$ . Wybierzmy dwie kolumny,  $c$  i  $c'$ , i niech  $x$  będzie dowolnym symbolem. Wówczas dla każdego symbolu  $x'$  istnieje jedyny wiersz macierzy  $A$ , w którym  $x$  występuje w kolumnie  $c$ , a  $x'$  w kolumnie  $c'$ . Przebiegając z  $x'$  cały zbiór  $X$ , widzimy, że  $x$  występuje w kolumnie  $c$  dokładnie  $n$  razy.

Ponieważ w pierwszym wierszu umieściliśmy same zera, tym samym wyczerpaliśmy wszystkie wystąpienia par  $(0, 0)$ . W rezultacie w żadnym innym wierszu nie ma dwóch zer. Policzmy wiersze, w których jest co najmniej jedno 0: jest ich  $1 + k(n - 1)$ . Macierz  $A$  ma  $n^2$  wierszy, zatem  $1 + k(n - 1) \leq n^2$  i w konsekwencji  $k \leq n + 1$ , co należało udowodnić. ■

Opiszymy teraz konstrukcję macierzy ortogonalnej dla  $\lambda = 1$ , w której  $k = n$ . W wyniku zastosowania tej konstrukcji powstała macierz ortogonalna z rysunku 10.5.

### TWIERDZENIE 10.7

Jeśli  $p$  jest liczbą pierwszą, to istnieje macierz ortogonalna  $\text{MO}(p, p, 1)$ .

**DOWÓD** Macierz, o której mowa w twierdzeniu, będzie macierzą wymiaru  $p^2 \times p$ , przy czym wiersze będą indeksowane elementami  $\mathbb{Z}_p \times \mathbb{Z}_p$ , a kolumny elementami  $\mathbb{Z}_p$ . Niech elementem stojącym w wierszu  $(i, j)$  i kolumnie  $x$  będzie  $ix + j \bmod p$ .

Wybierzmy dowolne dwie kolumny  $x, y$  ( $x \neq y$ ) i dwa symbole  $a, b$ . Chcemy znaleźć (jedyny) wiersz  $(i, j)$ , w którym  $a$  występuje w kolumnie  $x$ , a  $b$  w kolumnie  $y$  tego wiersza. Musimy zatem rozwiązać układ dwóch równań

$$a = ix + j$$

$$b = iy + j$$

z niewiadomymi  $i, j$  (działania są wykonywane w ciele  $\mathbb{Z}_p$ ). Ten układ ma jednoznaczne rozwiązanie:

$$i = (a - b)(x - y)^{-1} \bmod p$$

$$j = a - ix \bmod p.$$

Mamy zatem macierz ortogonalną. ■

Zauważmy, że każdą macierz  $\text{MO}(n, n, 1)$  można rozbudować przez dodanie jednej kolumny, tak aby otrzymać macierz  $\text{MO}(n, n + 1, 1)$  (patrz ćwiczenia). Dzięki twierdzeniu 10.7 można zatem otrzymać nieskończoną klasę macierzy ortogonalnych, dla których warunek z twierdzenia 10.6 jest spełniony ze znakiem równości.

Twierdzenie 10.6 mówi, że gdy  $k > n + 1$ , to  $\lambda > 1$ . Udowodnimy wynik ogólniejszy, ustalający dolne ograniczenie dla wartości  $\lambda$  w zależności od  $n$  i  $k$ . Zacznijmy jednak od wyprowadzenia ważnej nierówności, która się w zapowiadającym dowodzie przyda.

#### LEMAT 10.8

Niech  $b_1, \dots, b_m$  będą liczbami rzeczywistymi. Wówczas

$$m \sum_{i=1}^m b_i^2 \geq \left( \sum_{i=1}^m b_i \right)^2.$$

**DOWÓD** Użyjemy nierówności Jensen (twierdzenie 2.5) dla funkcji  $f(x) = -x^2$  ze współczynnikami  $a_i = 1/m$ ,  $1 \leq i \leq m$ . Funkcja  $f$  jest ciągła i wklęsła, zatem

$$-\sum_{i=1}^m \frac{b_i^2}{m} \leq -\left( \sum_{i=1}^m \frac{b_i}{m} \right)^2,$$

co po uproszczeniu daje żądaną nierówność. ■

#### TWIERDZENIE 10.9

Jeśli istnieje macierz ortogonalna  $\text{MO}(n, k, \lambda)$ , to

$$\lambda \geq \frac{k(n-1)+1}{n^2}.$$

**DOWÓD** Niech  $A$  będzie macierzą typu  $\text{MO}(n, k, \lambda)$  nad zbiorem symboli  $X = \{0, 1, \dots, n-1\}$ . Możemy założyć bez utraty ogólności, że pierwszy wiersz macierzy  $A$  składa się z samych zer (jak w twierdzeniu 10.6).

Niech  $\mathcal{R}$  oznacza zbiór wierszy macierzy  $A$ . Oznaczając pierwszy wiersz przez  $r_1$ , określmy zbiór  $\mathcal{R}_1 = \mathcal{R} \setminus \{r_1\}$ . Dla dowolnego wiersza  $r$  macierzy  $A$  niech  $x_r$  będzie liczbą wystąpień zera w wierszu  $r$ . Nietrudno obliczyć łączną liczbę wystąpień 0 w  $\mathcal{R}_1$ . Każdy symbol musi wystąpić w dowolnej kolumnie  $A$  dokładnie  $\lambda n$  razy, zatem

$$\sum_{r \in \mathcal{R}_1} x_r = k(\lambda n - 1).$$

Obliczmy teraz, ile razy w wierszach ze zbioru  $\mathcal{R}_1$  wystąpi para  $(0, 0)$ :

$$\begin{aligned}\sum_{r \in \mathcal{R}_1} x_r(x_r - 1) &= \sum_{r \in \mathcal{R}_1} x_r^2 - \sum_{r \in \mathcal{R}_1} x_r \\ &= \sum_{r \in \mathcal{R}_1} x_r^2 - k(\lambda n - 1).\end{aligned}$$

Na mocy lematu 10.8 otrzymujemy

$$\sum_{r \in \mathcal{R}_1} x_r^2 \geq \frac{(k(\lambda n - 1))^2}{\lambda n^2 - 1},$$

a stąd

$$\sum_{r \in \mathcal{R}_1} x_r(x_r - 1) \geq \frac{(k(\lambda n - 1))^2}{\lambda n^2 - 1} - k(\lambda n - 1).$$

Jednakże w dowolnie wybranej parze kolumn para  $(0, 0)$  występuje w dokładnie  $\lambda$  wierszach. Uporządkowanych par kolumn mamy  $k(k-1)$ , więc para  $(0, 0)$  występuje  $(\lambda-1)k(k-1)$  razy w wierszach ze zbioru  $\mathcal{R}_1$ , a w rezultacie

$$(\lambda-1)k(k-1) \geq \frac{(k(\lambda n - 1))^2}{\lambda n^2 - 1} - k(\lambda n - 1),$$

lub równoważnie,

$$((\lambda-1)k(k-1) + k(\lambda n - 1))(\lambda n^2 - 1) \geq (k(\lambda n - 1))^2.$$

Dzieląc obie strony przez  $k$ , mamy

$$(\lambda k - k - \lambda + \lambda n)(\lambda n^2 - 1) \geq k(\lambda n - 1)^2.$$

Rozwijamy obie strony:

$$\lambda^2 kn^2 - \lambda kn^2 - \lambda^2 n^2 + \lambda^2 n^3 - \lambda k + k + \lambda - \lambda n \geq \lambda^2 kn^2 - 2\lambda kn + k,$$

co po uproszczeniu daje

$$-\lambda^2 n^2 + \lambda^2 n^3 \geq \lambda kn^2 + \lambda k - \lambda + \lambda n - 2\lambda kn,$$

lub równoważnie

$$\lambda^2(n^3 - n^2) \geq \lambda(k(n-1)^2 + n + 1).$$

Na koniec podzielimy obie strony przez  $\lambda(n-1)$ . Otrzymujemy

$$\lambda n^2 \geq k(n-1) + 1$$

zgodnie z tezą twierdzenia. ■

Wykażemy teraz istnienie nieskończonej klasy macierzy ortogonalnych, dla których w udowodnionej wyżej nierówności zachodzi równość.

**TWIERDZENIE 10.10**

Niech  $p$  będzie liczbą pierwszą, a  $d \geq 2$  liczbą całkowitą. Wówczas istnieje macierz ortogonalna  $\text{MO}(p, (p^d - 1)/(p - 1), p^{d-2})$ .

**DOWÓD** Niech  $(\mathbb{Z}_p)^d$  oznacza przestrzeń liniową wszystkich ciągów  $d$ -wymiarowych nad  $\mathbb{Z}_p$ . Zbudujemy macierz  $A$  typu  $\text{MO}(p, (p^d - 1)/(p - 1), p^{d-2})$ , w której wiersze i kolumny są indeksowane pewnymi wektorami z przestrzeni  $(\mathbb{Z}_p)^d$ , a elementy należą do  $\mathbb{Z}_p$ . Zbiorem wierszy jest cała przestrzeń  $\mathcal{R} = (\mathbb{Z}_p)^d$ , natomiast zbiór kolumn określmy następująco:

$$\mathcal{C} = \{(c_1, \dots, c_d) \in (\mathbb{Z}_p)^d : \exists j, 0 \leq j \leq d-1, c_1 = \dots = c_j = 0, c_{j+1} = 1\}.$$

Zbiór  $\mathcal{R}$  składa się ze wszystkich wektorów przestrzeni  $(\mathbb{Z}_p)^d$ , więc  $|\mathcal{R}| = p^d$ . Z kolei  $\mathcal{C}$  składa się ze wszystkich niezerowych wektorów, których pierwszą niezerową współrzędną jest 1. Zauważmy, że

$$|\mathcal{C}| = \frac{p^d - 1}{p - 1},$$

a ponadto żadne dwa wektory należące do  $\mathcal{C}$  nie są wzajemnie swoimi wielokrotnościami skalarnymi.

Dalej, dla każdego  $\bar{r} \in \mathcal{R}$  i dla każdego  $\bar{c} \in \mathcal{C}$  definiujemy

$$A(\bar{r}, \bar{c}) = \bar{r} \cdot \bar{c},$$

gdzie kropka oznacza iloczyn skalarny obu wektorów (zredukowany modulo  $p$ ).

Udowodnimy teraz, że  $A$  jest żądaną macierzą. Niech  $\bar{b}, \bar{c} \in \mathcal{C}$  będą dwiema różnymi kolumnami i niech  $x, y \in \mathbb{Z}_p$ . Ustalimy liczbę wierszy  $\bar{r}$ , takich że  $A(\bar{r}, \bar{b}) = x$  i  $A(\bar{r}, \bar{c}) = y$ . Niech  $\bar{r} = (r_1, r_2, \dots, r_d)$ ,  $\bar{b} = (b_1, b_2, \dots, b_d)$  oraz  $\bar{c} = (c_1, c_2, \dots, c_d)$ . Równania  $\bar{r} \cdot \bar{b} = x$  i  $\bar{r} \cdot \bar{c} = y$  możemy zapisać w postaci dwóch równań liniowych nad  $\mathbb{Z}_p$ :

$$b_1 r_1 + \dots + b_d r_d = x$$

$$c_1 r_1 + \dots + c_d r_d = y.$$

Mamy tu układ dwóch równań liniowych z  $d$  niewiadomymi  $r_1, \dots, r_d$ . Z założenia para wektorów  $\bar{b}, \bar{c}$  jest liniowo niezależna, zatem niezależne są także oba równania. Wynika stąd, że przestrzeń rozwiązań jest wymiarem  $d-2$ , a więc rozwiązań (czyli wierszy, w których  $x$  występuje w kolumnie  $\bar{b}$ , a  $y$  w kolumnie  $\bar{c}$ ) jest  $p^{d-2}$ , zgodnie z tezą twierdzenia. ■

Popatrzmy na prosty przykład takiej konstrukcji.

**Przykład 10.3**

Przyjmijmy  $p = 2$ ,  $d = 3$ . Zbudujemy  $\text{MO}(2, 7, 2)$ . Mamy

$$\mathcal{R} = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

oraz

$$\mathcal{C} = \{001, 010, 011, 100, 101, 110, 111\}.$$

Wynikiem konstrukcji jest macierz przedstawiona na rysunku 10.6.  $\square$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

RYSUNEK 10.6. MO(2, 7, 2)

### 10.3.3. Charakteryzacje kodów uwierzytelniania

Badaliśmy dotąd kody uwierzytelniania otrzymane z macierzy ortogonalnych. Poznaliśmy warunki konieczne istnienia oraz konstrukcje macierzy ortogonalnych. Można zapytać, czy nie ma lepszych sposobów budowania kodów uwierzytelniania. Dwa twierdzenia charakteryzacyjne, które tu przedstawimy, mówią, że jeśli ograniczymy się do kodów uwierzytelniania z małymi prawdopodobieństwami oszustwa, to podejście związane z macierzami ortogonalnymi jest istotnie najlepsze.

Zaczniemy od udowodnienia częściowej odwrotności twierdzenia 10.5.

#### TWIERDZENIE 10.11

Niech  $(\mathcal{S}, \mathcal{A}, \mathcal{K}, \mathcal{E})$  będzie kodem uwierzytelniania, w którym  $|\mathcal{A}| = n$  oraz  $Pd_0 = Pd_1 = 1/n$ . Wówczas  $|\mathcal{K}| \geq n^2$ . Ponadto,  $|\mathcal{K}| = n^2$  wtedy i tylko wtedy, gdy istnieje macierz ortogonalna  $MO(n, k, 1)$ , w której  $|\mathcal{S}| = k$  oraz  $p_{\mathcal{K}}(K) = 1/n^2$  dla każdego klucza  $K \in \mathcal{K}$ .

**DOWÓD** Ustalmy dwa stany źródłowe  $s$  i  $s'$ ,  $s \neq s'$ , i rozważmy równanie (10.6). Dla każdej pary uporządkowanej  $(a, a')$  etykiet uwierzytelniających definiujemy

$$\mathcal{K}_{a,a'} = \{K \in \mathcal{K} : e_K(s) = a, e_K(s') = a'\}.$$

Wtedy  $|\mathcal{K}_{a,a'}| > 0$  dla każdej pary  $(a, a')$ . Ponadto zbiory  $\mathcal{K}_{a,a'}$ , których jest  $n^2$ , są rozłączne, zatem  $|\mathcal{K}| \geq n^2$ .

Załóżmy, że  $|\mathcal{K}| = n^2$ . Wtedy  $|\mathcal{K}_{a,a'}| = 1$  dla każdej pary  $(a, a')$ , a z równania (10.6) wynika, że  $p_{\mathcal{K}}(K) = 1/n^2$  dla każdego klucza  $K \in \mathcal{K}$ .

Pozostaje wykazać, że macierz uwierzytelniania jest macierzą ortogonalną typu  $\text{MO}(n, k, 1)$ . Popatrzmy na dwie kolumny indeksowane wybranymi stanami źródłowymi  $s$  i  $s'$ . Ponieważ  $|\mathcal{K}_{a,a'}| = 1$  dla dowolnej pary  $(a, a')$ , każda para uporządkowana występuje w tych kolumnach tylko jeden raz. Z dowolności wyboru  $s$  i  $s'$  wnioskujemy, że każda para uporządkowana występuje dokładnie jeden raz w dowolnej parze kolumn. ■

Następna charakteryzacja jest trudniejsza, dlatego przedstawiamy ją bez dowodu.

### TWIERDZENIE 10.12

Niech  $(\mathcal{S}, \mathcal{A}, \mathcal{K}, \mathcal{E})$  będzie kodem uwierzytelniania, w którym  $|\mathcal{S}| = k$ ,  $|\mathcal{A}| = n$  oraz  $Pd_0 = Pd_1 = 1/n$ . Wtedy  $|\mathcal{K}| \geq k(n - 1) + 1$ . Ponadto,  $|\mathcal{K}| = k(n - 1) + 1$  wtedy i tylko wtedy, gdy istnieje macierz ortogonalna  $\text{MO}(n, k, \lambda)$ , taka że  $\lambda = (k(n - 1) + 1)/n^2$ , oraz  $p_{\mathcal{K}}(K) = 1/(k(n - 1) + 1)$  dla każdego klucza  $K \in \mathcal{K}$ .

**UWAGA** Zauważmy, że twierdzenie 10.10 dostarcza nieskończonej klasy macierzy ortogonalnych, które spełniają warunek z twierdzenia 10.12 ze znakiem równości. ■

## 10.4. Ograniczenia związane z entropią

Użyjemy tu metod związanych z entropią do uzyskania ograniczeń na prawdopodobieństwa oszustwa. Pierwsze z nich dotyczy prawdopodobieństwa  $Pd_0$ .

### TWIERDZENIE 10.13

Niech  $(\mathcal{S}, \mathcal{A}, \mathcal{K}, \mathcal{E})$  będzie kodem uwierzytelniania. Wtedy

$$\log Pd_0 \geq H(\mathbf{K}|\mathbf{M}) - H(\mathbf{K}).$$

**DOWÓD** Z równania (10.1) mamy

$$Pd_0 = \max\{po(s, a) : s \in \mathcal{S}, a \in \mathcal{A}\}.$$

Maksimum wartości  $po(s, a)$  nie jest mniejsze niż ich średnia ważona, zatem otrzymujemy

$$Pd_0 \geq \sum_{s \in \mathcal{S}, a \in \mathcal{A}} p_{\mathcal{M}}(s, a)po(s, a).$$

Z nierówności Jensena (twierdzenie 2.5) mamy

$$\begin{aligned}\log Pd_0 &\geq \log \sum_{s \in \mathcal{S}, a \in \mathcal{A}} p_{\mathcal{M}}(s, a) po(s, a) \\ &\geq \sum_{s \in \mathcal{S}, a \in \mathcal{A}} p_{\mathcal{M}}(s, a) \log po(s, a).\end{aligned}$$

W podrozdziale 10.2 stwierdziliśmy, że

$$p_{\mathcal{M}}(s, a) = p_{\mathcal{S}}(s) \cdot po(s, a),$$

więc

$$\log Pd_0 \geq \sum_{s \in \mathcal{S}, a \in \mathcal{A}} p_{\mathcal{S}}(s) po(s, a) \log po(s, a).$$

Zauważmy teraz, że  $po(s, a) = p_{\mathcal{A}}(a|s)$  (prawdopodobieństwo tego, że  $a$  jest etykietą uwierzytelniającą dla stanu źródłowego  $s$ ). Stąd

$$\begin{aligned}\log Pd_0 &\geq \sum_{s \in \mathcal{S}, a \in \mathcal{A}} p_{\mathcal{S}}(s) p_{\mathcal{A}}(a|s) \log p_{\mathcal{A}}(a|s) \\ &= -H(\mathbf{A}|\mathbf{S}),\end{aligned}$$

na mocy definicji entropii warunkowej. Zakończymy dowód, wykazując, że  $-H(\mathbf{A}|\mathbf{S}) = H(\mathbf{K}|\mathbf{M}) - H(\mathbf{K})$ . Wynika to z podstawowych własności entropii. Z jednej strony mamy

$$H(\mathbf{K}, \mathbf{A}, \mathbf{S}) = H(\mathbf{K}|\mathbf{A}, \mathbf{S}) + H(\mathbf{A}|\mathbf{S}) + H(\mathbf{S}).$$

Z drugiej obliczamy

$$\begin{aligned}H(\mathbf{K}, \mathbf{A}, \mathbf{S}) &= H(\mathbf{A}|\mathbf{K}, \mathbf{S}) + H(\mathbf{K}, \mathbf{S}) \\ &= H(\mathbf{K}) + H(\mathbf{S}),\end{aligned}$$

korzystając z tego, że po pierwsze,  $H(\mathbf{A}|\mathbf{K}, \mathbf{S}) = 0$ , gdyż klucz i stan źródłowy jednoznacznie wyznaczają etykietę uwierzytelniającą, a po drugie, że  $H(\mathbf{K}, \mathbf{S}) = H(\mathbf{K}) + H(\mathbf{S})$ , gdyż stan źródłowy i klucz są zdarzeniami niezależnymi.

Przyrównując oba wyrażenia na  $H(\mathbf{K}, \mathbf{A}, \mathbf{S})$ , mamy

$$-H(\mathbf{A}|\mathbf{S}) = H(\mathbf{K}|\mathbf{A}, \mathbf{S}) - H(\mathbf{K}).$$

Jednakże wiadomość  $m = (s, a)$  z definicji składa się ze stanu źródłowego i etykiety uwierzytelniającej (a więc  $\mathcal{M} = \mathcal{S} \times \mathcal{A}$ ), zatem  $H(\mathbf{K}|\mathbf{A}, \mathbf{S}) = H(\mathbf{K}|\mathbf{M})$ , co kończy dowód. ■

Podobne ograniczenie dotyczy prawdopodobieństwa  $Pd_1$ , jednak nie będzie my tu przedstawić dowodu. Oto ono:

**TWIERDZENIE 10.14**

Niech  $(\mathcal{S}, \mathcal{A}, \mathcal{K}, \mathcal{E})$  będzie kodem uwierzytelniania. Wtedy

$$\log Pd_1 \geq H(\mathbf{K}|\mathbf{M}^2) - H(\mathbf{K}|\mathbf{M}).$$

Należy wyjaśnić, czym jest zmienna losowa  $\mathbf{M}^2$ . Przypuśćmy, że uwierzytelniamy dwa różne stanów źródłowe za pomocą tego samego klucza  $K$ . Otrzymujemy w ten sposób uporządkowaną parę wiadomości  $(m_1, m_2) \in \mathcal{M} \times \mathcal{M}$ . Aby zdefiniować rozkład prawdopodobieństwa na  $\mathcal{M} \times \mathcal{M}$ , musimy określić rozkład na  $\mathcal{S} \times \mathcal{S}$ , przyjmując przy tym, że  $p_{\mathcal{S} \times \mathcal{S}}(s, s) = 0$  dla każdego  $s \in \mathcal{S}$  (inaczej mówiąc, nie dopuszcamy powtarzania stanu źródłowego). Rozkłady prawdopodobieństwa na  $\mathcal{K}$  oraz na  $\mathcal{S} \times \mathcal{S}$  indukują rozkład na  $\mathcal{M} \times \mathcal{M}$  w taki sam sposób, jak rozkłady prawdopodobieństwa na  $\mathcal{K}$  i  $\mathcal{S}$  indukują rozkład na  $\mathcal{M}$ .

Aby zilustrować oba ograniczenia, powróćmy do naszej podstawowej konstrukcji macierzy ortogonalnej i wykażmy, że ograniczenia z twierdzeń 10.13 i 10.14 spełnione są ze znakiem równości. Przede wszystkim jest oczywiste, że

$$H(\mathbf{K}) = \log \lambda n^2,$$

ponieważ każdą z  $\lambda n^2$  reguł uwierzytelniania można wybrać z jednakowym prawdopodobieństwem. Zajmijmy się teraz obliczeniem  $H(\mathbf{K}|\mathbf{M})$ . Zaobserwowanie dowolnej wiadomości  $m = (s, a)$  ogranicza zbiór możliwych kluczy do podzbioru  $\lambda n$ -elementowego, przy czym każdy z tych  $\lambda n$  kluczy jest równie prawdopodobny. Stąd  $H(\mathbf{K}|m) = \log \lambda n$  dla dowolnej wiadomości  $m$ . Otrzymujemy zatem

$$\begin{aligned} H(\mathbf{K}|\mathbf{M}) &= \sum_{m \in \mathcal{M}} p_{\mathcal{M}}(m) H(\mathbf{K}|m) \\ &= \sum_{m \in \mathcal{M}} p_{\mathcal{M}}(m) \log \lambda n \\ &= \log \lambda n. \end{aligned}$$

W rezultacie mamy

$$H(\mathbf{K}|\mathbf{M}) - H(\mathbf{K}) = \log \lambda n - \log \lambda n^2 = -\log n = \log Pd_0,$$

co dowodzi, że ograniczenie górne określone w twierdzeniu 10.13 jest osiągane.

Gdy zaobserwujemy dwie wiadomości utworzone za pomocą tego samego klucza (z różnych stanów źródłowych), możemy ograniczyć liczbę możliwych kluczy do  $\lambda$ . Rozumując podobnie jak poprzednio, otrzymujemy  $H(\mathbf{K}|\mathbf{M}^2) = \log \lambda$ . Stąd

$$H(\mathbf{K}|\mathbf{M}^2) - H(\mathbf{K}|\mathbf{M}) = \log \lambda - \log \lambda n = -\log n = \log Pd_1,$$

a więc ograniczenie górne z twierdzenia 10.14 jest także osiągane.

### 10.5. Uwagi i bibliografia

Kody uwierzytelniania wymyślili Gilbert, MacWilliams i Sloane w 1974 roku [GMS74]. Teorię kodów uwierzytelniania opracował w znacznym stopniu Simmonds, który uzyskał wiele zasadniczych wyników dla tej dziedziny. Warto polecić dwa artykuły przeglądowe Simmonsa: [St92] i [St88]. Dobry przegląd zawiera również artykuł Masseya [MA86].

Wielu badaczy zajmowało się związkami między macierzami ortogonalnymi a kodami uwierzytelniania. Przedstawione podejście jest oparte na trzech pracach Stinsona [ST88], [ST90] oraz [ST92]. Od ponad 50 lat macierze ortogonalne są badane w ramach statystyki i kombinatoryki. Na przykład, ograniczenie z twierdzenia 10.9 zostało udowodnione po raz pierwszy przez Placketta i Burnama w 1945 roku [PB45]. Wiele ciekawych wyników dotyczących tych macierzy można znaleźć w podręcznikach z kombinatorycznej matematyki dyskretnej, na przykład w książce Betha, Jungnickela i Lenza [BJL85].

Zastosowanie metod związanych z entropią do badania kodów uwierzytelniania zawdzięczamy Simmonsovi. Pierwszy dowód ograniczenia z twierdzenia 10.13 zawiera [St85], dowód twierdzenia 10.14 można znaleźć w pracy Walkera [WA90].

### Ćwiczenia

- 10.1. Oblicz  $Pd_0$  i  $Pd_1$  dla następujących kodów uwierzytelniania przedstawionych w postaci macierzowej:

| Klucz | 1 | 2 | 3 | 4 |
|-------|---|---|---|---|
| 1     | 1 | 1 | 2 | 3 |
| 2     | 1 | 2 | 3 | 1 |
| 3     | 2 | 1 | 3 | 1 |
| 4     | 2 | 3 | 1 | 2 |
| 5     | 3 | 2 | 1 | 3 |
| 6     | 3 | 3 | 2 | 1 |

Rozkłady prawdopodobieństwa na  $\mathcal{S}$  i  $\mathcal{K}$  określone są następująco:

$$\begin{aligned} p_{\mathcal{S}}(1) &= p_{\mathcal{S}}(4) = 1/6 \\ p_{\mathcal{S}}(2) &= p_{\mathcal{S}}(3) = 1/3 \\ p_{\mathcal{K}}(1) &= p_{\mathcal{K}}(6) = 1/4 \\ p_{\mathcal{K}}(2) &= p_{\mathcal{K}}(3) = p_{\mathcal{K}}(4) = p_{\mathcal{K}}(5) = 1/8. \end{aligned}$$

Jakie strategie podszywania się i podstawienia są tu optymalne?

- 10.2. Omawialiśmy wyżej konstrukcję macierzy ortogonalnej  $MO(p, p, 1)$ , gdy  $p$  jest liczbą pierwszą. Wykaż, że taką macierz  $MO(p, p, 1)$  można zawsze rozszerzyć o jedną kolumnę do macierzy  $MO(p, p+1, 1)$ . Przeprowadź taką konstrukcję dla  $p = 5$ .

- 10.3. Niech  $A$  będzie macierzą ortogonalną typu  $\text{MO}(n_1, k, \lambda_1)$  nad zbiorem symboli  $\{1, \dots, n_1\}$  i niech  $B$  będzie macierzą ortogonalną typu  $\text{MO}(n_2, k, \lambda_2)$  nad zbiorem symboli  $\{1, \dots, n_2\}$ . Zbudujmy macierz  $C$  typu  $\text{MO}(n_1 n_2, k, \lambda_1 \lambda_2)$  nad zbiorem symboli  $\{1, \dots, n_1\} \times \{1, \dots, n_2\}$ : dla każdego wiersza  $r_1 = (x_1, \dots, x_k)$  macierzy  $A$  i wiersza  $s_1 = (y_1, \dots, y_k)$  macierzy  $B$  definiujemy wiersz

$$t_1((x_1, y_1), \dots, (x_k, y_k))$$

macierzy  $C$ . Wykaż, że  $C$  jest rzeczywiście macierzą typu  $\text{MO}(n_1 n_2, k, \lambda_1 \lambda_2)$ .

- 10.4. Zbuduj macierz ortogonalną  $\text{MO}(3, 13, 3)$ .

- 10.5. Napisz program komputerowy do obliczenia wartości  $H(\mathbf{K})$ ,  $H(\mathbf{K}|\mathbf{M})$  oraz  $H(\mathbf{K}|\mathbf{M}^2)$  dla kodu uwierzytelniania z ćwiczenia 10.1, jeśli rozkład prawdopodobieństwa dla par stanów źródłowych jest następujący:

$$p_{S^2}(1, 2) = p_{S^2}(1, 3) = p_{S^2}(1, 4) = 1/18$$

$$p_{S^2}(2, 1) = p_{S^2}(2, 3) = p_{S^2}(2, 4) = 1/9$$

$$p_{S^2}(3, 1) = p_{S^2}(3, 2) = p_{S^2}(3, 4) = 1/9$$

$$p_{S^2}(4, 1) = p_{S^2}(4, 2) = p_{S^2}(4, 3) = 1/18$$

Porównaj związane z entropią ograniczenia na  $Pd_0$  i  $Pd_1$  z rzeczywistymi wartościami obliczonymi w ćwiczeniu 10.1.

**WSKAZÓWKA** Do obliczenia  $p_{\mathcal{K}}(k|m)$  użyj wzoru Bayesa:

$$p_{\mathcal{K}}(k|m) = \frac{p_{\mathcal{M}}(m|k)p_{\mathcal{K}}(k)}{p_{\mathcal{M}}(m)}.$$

Wiemy już, jak obliczyć  $p_{\mathcal{M}}(m)$ . Zauważmy, że jeśli  $m = (s, a)$ , to  $p_{\mathcal{M}}(m|k) = p_{\mathcal{S}}(s)$ , gdy  $e_K(s) = a$ , i  $p_{\mathcal{M}}(m|k) = 0$  w pozostałych przypadkach.

Do obliczenia  $p_{\mathcal{K}}(k|m_1, m_2)$  należy ponownie użyć wzoru Bayesa:

$$p_{\mathcal{K}}(k|m_1, m_2) = \frac{p_{\mathcal{M}^2}(m_1, m_2|k)p_{\mathcal{K}}(k)}{p_{\mathcal{M}^2}(m_1, m_2)}.$$

Prawdopodobieństwo  $p_{\mathcal{M}^2}(m_1, m_2)$  można obliczyć w sposób następujący: niech  $m_1 = (s_1, a_1)$  i  $m_2 = (s_2, a_2)$ . Wtedy

$$p_{\mathcal{M}^2}(m_1, m_2) = p_{S^2}(s_1, s_2) \times \sum_{\{K \in \mathcal{K}: e_K(s_1)=a_1, e_K(s_2)=a_2\}} p_{\mathcal{K}}(K).$$

(Zwrócić uwagę na podobieństwo do obliczenia wartości  $p(m)$ ). By obliczyć  $p_{\mathcal{M}^2}(m_1, m_2|k)$ , zauważ, że  $p_{\mathcal{M}^2}(m_1, m_2|k) = p_{S^2}(s_1, s_2)$ , gdy  $e_K(s_1) = a_1$  i  $e_K(s_2) = a_2$ , a  $p_{\mathcal{M}^2}(m_1, m_2|k) = 0$  w pozostałych przypadkach.

# 11

---

## Tajne schematy współużytkowania

---

### 11.1. Wprowadzenie: schemat progowy Shamira

Jeden z sejfów bankowych musi być codziennie otwierany. Bank zatrudnia trzech upoważnionych do tego pracowników, nie chce jednak, by którykolwiek z nich mógł sam dysponować otwierającą kombinacją. Potrzebny mu jest zatem system umożliwiający otwarcie sejfu dowolnym dwóm upoważnionym pracownikom, a jednocześnie nie dopuszczający do tego, by jeden z nich mógł to zrobić samodzielnie. Rozwiązaniem tego problemu jest *tajny schemat współużytkowania* (ang. *secret sharing scheme*), któremu poświęcony jest ten rozdział.

Oto ciekawy przykład tego rodzaju sytuacji z „rzeczywistego świata”: według czasopisma *Time Magazine*<sup>1</sup> zarządzanie bronią jądrową w Rosji opiera się na podobnym mechanizmie dostępu „dwóch spośród trzech”. Ową trójkę stanowią: prezydent, minister obrony oraz Ministerstwo Obrony.

Zaczniemy od omówienia pewnego szczególnego tajnego schematu współużytkowania, zwanego schematem progowym. Oto nieformalna definicja.

#### DEFINICJA 11.1

Niech  $t, w$  będą dodatnimi liczbami całkowitymi, takimi że  $t \leq w$ . *Schematem progowym o  $w$  użytkownikach i progu  $t$*  (w skrócie *schematem progowym o parametrach  $t, w$* ) jest taka metoda dzielenia klucza  $K$  w zbiorze  $w$  uczestników (oznaczanym  $\mathcal{P}$ ), która pozwala obliczyć wartość klucza  $K$  dowolnej grupie  $t$  uczestników, ale nie może tego uczynić żadna grupa  $t - 1$  uczestników.

Opisane wyżej sytuacje odnoszą się więc do (2,3)-schematów progowych.

Wartość klucza  $K$  określa specjalny uczestnik zwany *rozdającym* i oznaczany symbolem  $D$ . Zakładamy, że  $D \notin \mathcal{P}$ . Gdy  $D$  chce rozdzielić klucz  $K$  wśród uczestników ze zbioru  $\mathcal{P}$ , każdemu z nich podaje pewną częściową informację, zwaną *udziałem*. Udziały powinny być rozdzielone w sposób tajny, tak by żaden z uczestników nie znał udziałów przekazanych innym uczestnikom.

---

<sup>1</sup> *Time Magazine*, 4 maja 1992, s. 13.

Załóżmy, że po rozdzieleniu udziałów uczestnicy z pewnego podzbioru  $B \subseteq \mathcal{P}$  chcą połączyć swoje udziały w celu obliczenia klucza  $K$  (lub przekazują swoje udziały zaufanej osobie, aby wykonała obliczenia w ich imieniu). Jeśli  $|B| \geq t$ , znajomość posiadanych przez nich udziałów powinna wystarczyć do obliczenia wartości klucza  $K$ ; gdy  $|B| < t$ , obliczenie klucza powinno być niemożliwe.

Wprowadźmy pewne oznaczenia. Niech

$$\mathcal{P} = \{P_i : 1 \leq i \leq w\}$$

będzie zbiorem  $w$  uczestników,  $\mathcal{K}$  – zbiorem kluczy (czyli zbiorem wszystkich możliwych kluczy), a  $\mathcal{S}$  – zbiorem udziałów (czyli zbiorem wszystkich możliwych udziałów).

Opiszemy tu pochodzącą z 1979 roku metodę konstrukcji schematu progowego o parametrach  $t, w$ , zwanego **schematem progowym Shamira**. Niech  $\mathcal{K} = \mathcal{S} = \mathbb{Z}_p$ , gdzie  $p \geq w + 1$  jest liczbą pierwszą. Tak więc klucz, podobnie jak udział każdego z uczestników, jest elementem  $\mathbb{Z}_p$ . Schemat progowy Shamira przedstawiamy na rysunku 11.1. Zgodnie z nim rozdający tworzy losowy wielomian  $a(x)$  stopnia co najwyżej  $t - 1$ , w którym wyraz stały jest kluczem  $K$ . Każdy z uczestników  $P_i$  dostaje informację w postaci punktu  $(x_i, y_i)$  należącego do wykresu tego wielomianu.

### Faza inicjowania

- (1)  $D$  wybiera  $w$  różnych niezerowych elementów  $x_i \in \mathbb{Z}_p$  dla  $1 \leq i \leq w$  (dla tego wymagamy, by  $p \geq w + 1$ );  $D$  przekazuje wartość  $x_i$  do  $P_i$ , dla  $1 \leq i \leq w$ ; wartości te są znane publicznie

### Rozdanie udziałów

- (2) założmy, że  $D$  chce rozdać klucz  $K \in \mathbb{Z}_p$ ; wybiera potajemnie (losowo i niezależnie)  $t - 1$  elementów  $\mathbb{Z}_p$ :  $a_1, \dots, a_{t-1}$
- (3)  $D$  oblicza  $y_i = a(x_i)$  dla  $1 \leq i \leq w$ , gdzie

$$a(x) = K + \sum_{j=1}^{t-1} a_j x^j \pmod{p}$$

- (4)  $D$  przekazuje udział  $y_i$  do  $P_i$ , dla  $1 \leq i \leq w$

**RYSUNEK 11.1.** Schemat progowy Shamira w  $\mathbb{Z}_p$  z parametrami  $t, w$

Zobaczmy teraz, jak podzióbior  $B$  złożony z  $t$  uczestników może odtworzyć klucz. Zasadniczo odbywa się to w drodze interpolacji wielomianowej. Przedstawiemy kilka sposobów jej przeprowadzenia.

Załóżmy, że uczestnicy  $P_{i_1}, \dots, P_{i_t}$  chcą ustalić klucz  $K$ . Wiedzą oni, że

$$y_{i_j} = a(x_{i_j})$$

dla  $1 \leq j \leq t$ , gdzie  $a(x) \in \mathbb{Z}_p[x]$  jest (tajnym) wielomianem wybranym przez  $D$ . Wielomian  $a(x)$  ma stopień nie większy niż  $t-1$ , można go więc zapisać w postaci

$$a(x) = a_0 + a_1x + \dots + a_{t-1}x^{t-1},$$

gdzie współczynniki  $a_0, \dots, a_{t-1}$  są nieznanymi elementami  $\mathbb{Z}_p$ , a  $a_0 = K$  jest kluczem. Dla  $1 \leq j \leq t$  zachodzą równości  $y_{i_j} = a(x_{i_j})$ , zatem uczestnicy ze zbioru  $B$  otrzymują układ  $t$  równań z  $t$  niewiadomymi  $a_0, \dots, a_{t-1}$  nad ciałem  $\mathbb{Z}_p$ . Jeśli równania są liniowo niezależne, układ ma jednoznaczne rozwiązanie, którego składnikiem jest wartość klucza  $a_0$ .

Popatrzmy na prosty przykład.

### Przykład 11.1

Niech  $p = 17$ ,  $t = 3$  oraz  $w = 5$ . Publicznie znanymi odciętymi punktów niech będą  $x_i = i$ ,  $1 \leq i \leq 5$ . Przypuśćmy, że  $B = \{P_1, P_3, P_5\}$  jest zbiorem uczestników, którzy postanowili połączyć swoje udziały, równe odpowiednio 8, 10 i 11. Zapisując wielomian  $a(x)$  w postaci

$$a(x) = a_0 + a_1x + a_2x^2$$

i obliczając  $a(1)$ ,  $a(3)$  oraz  $a(5)$ , otrzymujemy układ trzech równań liniowych nad  $\mathbb{Z}_{17}$ :

$$a_0 + a_1 + a_2 = 8$$

$$a_0 + 3a_1 + 9a_2 = 10$$

$$a_0 + 5a_1 + 8a_2 = 11.$$

Układ ten ma jednoznaczne rozwiązanie w  $\mathbb{Z}_{17}$ :  $a_0 = 13$ ,  $a_1 = 10$  oraz  $a_2 = 2$ . Kluczem jest zatem  $K = a_0 = 13$ .  $\square$

Rzecz jasna układ  $t$  równań liniowych powinien mieć jednoznaczne rozwiązanie, tak jak w przykładzie 11.1. Wykażemy teraz, że tak jest zawsze. W ogólnym przypadku mamy

$$y_{i_j} = a(x_{i_j})$$

dla  $1 \leq j \leq t$ , gdzie

$$a(x) = a_0 + a_1x + \dots + a_{t-1}x^{t-1}$$

oraz

$$a_0 = K.$$

Układ równań liniowych (nad  $\mathbb{Z}_p$ ) wygląda zatem następująco:

$$a_0 + a_1x_{i_1} + a_2x_{i_1}^2 + \dots + a_{t-1}x_{i_1}^{t-1} = y_{i_1}$$

$$a_0 + a_1x_{i_2} + a_2x_{i_2}^2 + \dots + a_{t-1}x_{i_2}^{t-1} = y_{i_2}$$

$\vdots$

$$a_0 + a_1x_{i_t} + a_2x_{i_t}^2 + \dots + a_{t-1}x_{i_t}^{t-1} = y_{i_t}.$$

Można to zapisać krócej w postaci macierzowej:

$$\begin{pmatrix} 1 & x_{i_1} & {x_{i_1}}^2 & \dots & {x_{i_1}}^{t-1} \\ 1 & x_{i_2} & {x_{i_2}}^2 & \dots & {x_{i_2}}^{t-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{i_t} & {x_{i_t}}^2 & \dots & {x_{i_t}}^{t-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{pmatrix} = \begin{pmatrix} y_{i_1} \\ y_{i_2} \\ \vdots \\ y_{i_t} \end{pmatrix},$$

Macierz współczynników  $A$  jest tu tak zwaną macierzą Vandermonde'a. Istnieje dobrze znany wzór na wyznacznik takiej macierzy, mianowicie

$$\det A = \prod_{1 \leq j < k \leq t} (x_{i_k} - x_{i_j}) \bmod p.$$

Przypomnijmy, że wszystkie wartości  $x_i$  są różne, zatem żaden z czynników postaci  $x_{i_k} - x_{i_j}$  nie jest równy zeru. Obliczenia wykonywane są w  $\mathbb{Z}_p$ , a więc w ciele, w którym  $p$  jest liczbą pierwszą. Ponieważ w ciele iloczyn czynników różnych od 0 jest także różny od 0, zatem  $\det A \neq 0$ , a to oznacza, że układ ma jednoznaczne rozwiązanie w ciele  $\mathbb{Z}_p$ . Innymi słowy, dowolna grupa  $t$  uczestników potrafi w takim schemacie progowym odtworzyć wartość klucza.

Co się dzieje, gdy próbę odtworzenia klucza podejmie grupa  $t - 1$  uczestników? Postępując w opisany wyżej sposób, otrzymają oni układ  $t - 1$  równań z  $t$  niewiadomymi. Przypuśćmy, że przyjmą oni hipotetyczną wartość klucza  $y_0$ . Ponieważ kluczem jest  $a_0 = a(0)$ , wynika stąd kolejne równanie, a macierz tak rozszerzonego układu  $t$  równań z  $t$  niewiadomymi jest znowu macierzą Vandermonde'a. Oznacza to, że układ ma jednoznaczne rozwiązanie, a więc dla dowolnej hipotetycznej wartości klucza  $y_0$  istnieje jedyny wielomian  $a_{y_0}(x)$ , taki że

$$y_{i_j} = a_{y_0}(x_{i_j})$$

dla  $1 \leq j \leq t - 1$  oraz

$$y_0 = a_{y_0}(0).$$

Nie można więc wykluczyć żadnej wartości klucza, a to oznacza, że grupa  $t - 1$  uczestników nie jest w stanie uzyskać żadnej informacji o kluczu.

Analizując schemat Shamira, skupiliśmy się na rozwiążalności układów równań liniowych nad  $\mathbb{Z}_p$ . Można jednak przyjąć inny punkt wyjścia, związany z wielomianowym wzorem interpolacyjnym Lagrange'a. Wzór interpolacyjny Lagrange'a określa wprost obliczony wyżej (jedyny) wielomian  $a(x)$  stopnia co najwyżej  $t - 1$ . Oto ten wzór:

$$a(x) = \sum_{j=1}^t y_{i_j} \prod_{1 \leq k \leq t, k \neq j} \frac{x - x_{i_k}}{x_{i_j} - x_{i_k}}.$$

Łatwo sprawdzić poprawność tego wzoru, podstawiając  $x = x_{i_j}$ : znikają wszystkie składniki sumy oprócz  $y_{i_j}$ . Mamy zatem wielomian stopnia co najwyżej  $t - 1$ , do którego należą wszystkie pary uporządkowane  $(x_{i_j}, y_{i_j})$ ,  $1 \leq j \leq t$ .

Udowodniliśmy już, że taki wielomian jest wyznaczony jednoznacznie, zatem ze wzoru interpolacyjnego rzeczywiście otrzymujemy poszukiwany wielomian.

Grupa  $B$  złożona z  $t$  uczestników jest w stanie obliczyć wielomian  $a(x)$  za pomocą wzoru interpolacyjnego, ale może zrobić to także w sposób nieco prostszy. Uczestnikom z grupy  $B$  nie jest przecież potrzebny cały wielomian  $a(x)$ , a jedynie jego wyraz stały  $K = a_0$ . Mogą zatem obliczyć jedynie wyrażenie otrzymane ze wzoru interpolacyjnego po podstawieniu 0 za zmienną  $x$ :

$$K = \sum_{j=1}^t y_{i_j} \prod_{1 \leq k \leq t, k \neq j} \frac{x_{i_k}}{x_{i_k} - x_{i_j}}.$$

Niech

$$b_j = \prod_{1 \leq k \leq t, k \neq j} \frac{x_{i_k}}{x_{i_k} - x_{i_j}},$$

$1 \leq j \leq t$ . (Zauważmy, że wartości  $b_j$  można obliczyć zawsze i nie są one tajne). Mamy wówczas

$$K = \sum_{j=1}^t b_j y_{i_j}.$$

Tak więc klucz jest kombinacją liniową  $t$  udziałów.

Dla ilustracji opisanego tu podejścia obliczymy raz jeszcze tą metodą klucz z przykładu 11.1.

*Przykład 11.1 cd.*

Używając podanych wyżej wzorów, uczestnicy  $P_1, P_3, P_5$  mogą obliczyć  $b_1, b_2$  i  $b_3$ . Na przykład,

$$\begin{aligned} b_1 &= \frac{x_3 x_5}{(x_3 - x_1)(x_5 - x_1)} \bmod 17 \\ &= 3 \cdot 5 \cdot (-2)^{-1} \cdot (-4)^{-1} \bmod 17 \\ &= 4. \end{aligned}$$

Podobnie,  $b_2 = 3$  oraz  $b_3 = 11$ . Mając udziały 8, 10 i 11, odpowiednio, uczestnicy obliczają wartość klucza:

$$K = 4 \cdot 8 + 3 \cdot 10 + 11 \cdot 11 \bmod 17 = 13,$$

jak poprzednio. □

Na koniec tego podrozdziału omówimy uproszczoną konstrukcję schematu progowego w szczególnym przypadku, gdy  $w = t$ . Można ją przeprowadzić dla dowolnego zbioru kluczy  $\mathcal{K} = \mathbb{Z}_m$ , gdy  $\mathcal{S} = \mathbb{Z}_m$ . (W tym schemacie  $m$  nie musi być liczbą pierwszą, nie musi też być spełniona nierówność  $m \geq w + 1$ ). W celu

(1)  $D$  potajemnie wybiera (losowo i niezależnie)  $t - 1$  elementów  $\mathbb{Z}_m$ :

$$y_1, \dots, y_{t-1}$$

(2)  $D$  oblicza

$$y_t = K - \sum_{i=1}^{t-1} y_i \bmod m$$

(3)  $D$  przekazuje udział  $y_i$  do  $P_i$ , dla  $1 \leq i \leq t$

**RYSUNEK 11.2.** Schemat progowy w ciele  $\mathbb{Z}_m$  o parametrach  $(t, t)$

rozdzielenia klucza  $K \in \mathbb{Z}_m$  rozdający  $D$  korzysta z protokołu opisanego na rysunku 11.2.

Zauważmy, że  $t$  uczestników może obliczyć  $K$  za pomocą wzoru

$$K = \sum_{i=1}^t y_i \bmod m.$$

Czy  $t - 1$  uczestników jest w stanie obliczyć  $K$ ? Oczywiście, pierwszych  $t - 1$  uczestników tego zrobić nie może, gdyż otrzymują oni udziały w postaci  $t - 1$  niezależnych liczb losowych. Rozważmy teraz  $t - 1$  uczestników z grupy  $\mathcal{P} \setminus \{P_i\}$ , gdzie  $1 \leq i \leq t - 1$ . Mają oni następujące udziały:

$$y_1, \dots, y_{i-1}, y_{i+1}, \dots, y_{t-1}$$

oraz

$$K - \sum_{i=1}^{t-1}.$$

Dodając swoje udziały, mogą obliczyć  $K - y_i$ . Nie znając jednak wartości  $y_i$ , nie potrafią uzyskać na tej podstawie żadnej informacji o wartości  $K$ . Mamy więc do czynienia ze schematem progowym o parametrach  $t, t$ .

## 11.2. Struktury dostępu i ogólne tajne współużytkowanie

Rozpatrywaliśmy dotąd sytuację opartą na założeniu, że dowolna grupa  $t$  spośród  $w$  uczestników może obliczyć klucz. Uogólniając to założenie, możemy rozpatrywać przypadek, gdy niektóre grupy uczestników mają dostęp do klucza, a inne nie. Niech  $\Gamma$  będzie rodziną podzbiorów zbioru  $\mathcal{P}$ ; podzbiory należące do  $\Gamma$  będą właśnie tymi podzbiorami uczestników, dla których obliczenie klucza jest możliwe. Rodzinę  $\Gamma$  nazwiemy *strukturą dostępu*, a podzbiory należące do  $\Gamma$  – *podzbiorami upoważnionymi*.

Niech  $\mathcal{K}$  będzie zbiorem kluczy, a  $\mathcal{S}$  zbiorem udziałów. Jak poprzednio, gdy rozdający  $D$  chce rozdzielić klucz  $K \in \mathcal{K}$ , przydziela każdemu uczestnikowi udział ze zbioru  $\mathcal{S}$ . Łącząc swoje udziały, grupy uczestników mogą próbować ustalić klucz  $K$ , który wspólnie wykorzystują.

### DEFINICJA 11.2

*Doskonałym tajnym schematem współużytkowania* realizującym strukturę dostępu  $\Gamma$  nazywamy każdą metodę dzielenia klucza  $K$  w zbiorze  $w$  uczestników (oznaczonym  $\mathcal{P}$ ), która ma następujące własności:

- (1) uczestnicy z dowolnego upoważnionego podzbioru  $B \subseteq \mathcal{P}$  mogą obliczyć klucz  $K$ , łącząc swoje udziały;
- (2) uczestnicy nieupoważnionego podzbioru  $B \subseteq \mathcal{P}$  nie mogą, znając wszystkie swoje udziały, uzyskać żadnych informacji o kluczu  $K$ .

Zauważmy, że schemat progowy o parametrach  $t, w$  realizuje strukturę dostępu

$$\{B \subseteq \mathcal{P} : |B| \geq t\}.$$

Taką strukturę dostępu będziemy nazywać *strukturą progową*. Wykazaliśmy w poprzednim podrozdziale, że schemat Shamira jest doskonałym schematem realizującym progową strukturę dostępu.

Teraz zbadamy bezwarunkowe bezpieczeństwo tajnych schematów współużytkowania. Oznacza to, że nie nakładamy żadnych ograniczeń na liczbę obliczeń wykonywanych przez nieupoważniony podzbiór uczestników.

Załóżmy, że  $B \in \Gamma$  oraz  $B \subseteq C \subseteq \mathcal{P}$  i przypuśćmy, że podzbiór  $C$  chce ustalić klucz  $K$ . Uczestnicy z upoważnionego podzbioru  $B$  mogą sami obliczyć  $K$ , zatem to samo mogą zrobić członkowie grupy  $C$ , jako że mogą po prostu pominać udziały ze zbioru  $C \setminus B$ . Wynika stąd, że nadzbiór zbioru upoważnionego jest znowu zbiorem upoważnionym. Oznacza to, że struktura dostępu musi spełniać warunek *monotoniczności*:

jeśli  $B \in \Gamma$  oraz  $B \subseteq C \subseteq \mathcal{P}$ , to  $C \in \Gamma$ .

W pozostałej części tego rozdziału będziemy zakładać, że wszystkie struktury dostępu są monotoniczne.

Jeśli  $\Gamma$  jest strukturą dostępu, to podzbiór  $B \in \Gamma$  nazwiemy *minimalnym* podziobrem upoważnionym, gdy  $A \notin \Gamma$  dla dowolnego zbioru  $A \subseteq B$ ,  $A \neq B$ . Rodzinę wszystkich minimalnych upoważnionych podzbiorów struktury  $\Gamma$  nazywiemy *bazą*  $\Gamma$  i oznaczamy symbolem  $\Gamma_0$ . Ponieważ  $\Gamma$  składa się ze wszystkich zbiorów z rodziny  $\mathcal{P}$ , które są nadzbiorami zbiorów z bazy  $\Gamma_0$ , zatem  $\Gamma_0$  jednoznacznie określa strukturę  $\Gamma$ . Bardziej matematycznie,

$$\Gamma = \{C \subseteq \mathcal{P} : B \subseteq C, B \in \Gamma_0\}.$$

Mówimy wtedy, że  $\Gamma$  jest *domknięciem*  $\Gamma_0$  i piszemy

$$\Gamma = cl(\Gamma_0).$$

### Przykład 11.2

Niech  $\mathcal{P} = \{P_1, P_2, P_3, P_4\}$  oraz

$$\Gamma_0 = \{\{P_1, P_2, P_4\}, \{P_1, P_3, P_4\}, \{P_2, P_3\}\}.$$

Wtedy

$$\Gamma = \Gamma_0 \cup \{\{P_1, P_2, P_3\}, \{P_2, P_3, P_4\}, \{P_1, P_2, P_3, P_4\}\}.$$

I na odwrót, mając taką strukturę dostępu  $\Gamma$ , łatwo sprawdzić, że  $\Gamma_0$  składa się z minimalnych podzbiorów rodziny  $\Gamma$ .  $\square$

W przypadku progowej struktury dostępu o parametrach  $t, w$ , bazę stanowią wszystkie podzbiory składające się z (dokładnie)  $t$  uczestników.

## 11.3. Konstrukcja oparta na obwodzie monotonicznym

Podamy tu pojęciowo prostą i elegancką konstrukcję Benaloha i Leichtera pokazującą, że każdą (monotoniczną) strukturę dostępu można zrealizować w pewnym doskonałym tajnym schemacie współużytkowania. Istota pomysłu polega na tym, że najpierw buduje się monotoniczny obwód, który „rozpozna” daną strukturę dostępu, a następnie tworzy się tajny schemat współużytkowania na podstawie opisu tego obwodu. Tę metodę nazwiemy *konstrukcją opartą na obwodzie monotonicznym* (ang. *monotone circuit construction*).

Przypuśćmy, że mamy obwód boolowski  $\mathbf{C}$  z  $w$  wejściami boolowskimi  $x_1, \dots, x_w$  (odpowiadającymi  $w$  uczestnikom  $P_1, \dots, P_w$ ) oraz jedno boolowskie wyjście  $y$ . Obwód składa się z bramek OR i AND, nie dopuszczamy natomiast bramek NOT. Taki obwód nazwiemy obwodem *monotonicznym*, gdyż zmieniając wartość na wejściu z „0” (fałsz) na „1” (prawda), nigdy nie spowodujemy zmiany wartości na wyjściu z „1” na „0”. Obwód może mieć dowolną obciążalność wejściową, ale obciążalność wyjściowa może być równa tylko 1 (oznacza to, że bramka może mieć dowolnie dużo wejść, ale tylko jedno wyjście).

Gdy określmy wartości boolowskie dla każdego z  $w$  wejść takiego obwodu monotonicznego, możemy określić zbiór

$$B(x_1, \dots, x_w) = \{P_i : x_i = 1\},$$

czyli podzbiór zbioru  $\mathcal{P}$  odpowiadający wejściom o wartości „1”. Niech  $\mathbf{C}$  będzie obwodem monotonicznym; określmy

$$\Gamma(\mathbf{C}) = \{B(x_1, \dots, x_w) : \mathbf{C}(x_1, \dots, x_w) = 1\},$$

gdzie  $\mathbf{C}(x_1, \dots, x_w)$  oznacza wyjście obwodu  $\mathbf{C}$  odpowiadające wejściom  $x_1, \dots, x_w$ . Z monotoniczności obwodu  $\mathbf{C}$  wynika, że  $\Gamma(\mathbf{C})$  jest monotoniczną rodziną podzbiorów zbioru  $\mathcal{P}$ .

Łatwo zauważyc, że istnieje wzajemnie jednoznaczna odpowiedniość między obwodami monotonicznymi tego typu a formułami boolowskimi zawierającymi wyłącznie spójniki  $\wedge$  (AND) i  $\vee$  (OR), nie zawierającymi zaś żadnej negacji.

Jeśli  $\Gamma$  jest monotoniczną rodziną podzbiorów zbioru  $\mathcal{P}$ , to bez trudu można zbudować obwód monotoniczny  $\mathbf{C}$ , taki że  $\Gamma(\mathbf{C}) = \Gamma$ . Na przykład, niech  $\Gamma_0$  będzie bazą  $\Gamma$ . Budujemy wtedy formułę boolowską w *dysjunktywnej postaci normalnej*:

$$\bigvee_{B \in \Gamma_0} \left( \bigwedge_{P_i \in B} P_i \right).$$

W przykładzie 11.2, gdzie

$$\Gamma_0 = \{\{P_1, P_2, P_4\}, \{P_1, P_3, P_4\}, \{P_2, P_3\}\},$$

otrzymalibyśmy formułę

$$(P_1 \wedge P_2 \wedge P_4) \vee (P_1 \wedge P_3 \wedge P_4) \vee (P_2 \wedge P_3). \quad (11.1)$$

Każdy składnik takiej formuły boolowskiej odpowiada bramce AND w odpowiednim obwodzie monotonicznym; cała alternatywa odpowiada bramce OR. Obwód liczy zatem  $|\Gamma_0| + 1$  bramek.

Niech  $\mathbf{C}$  będzie obwodem monotonicznym rozpoznającym  $\Gamma$  (zauważmy, że  $\mathbf{C}$  nie musi być obwodem opisanym wyżej). Przedstawimy algorytm umożliwiający rozdającemu  $D$  zbudowanie doskonałego tajnego schematu współużytkowania realizującego  $\Gamma$ . Składowymi tego schematu będą schematy o parametrach  $t, t$ , zbudowane jak na rysunku 11.2. Dlatego przyjmujemy, że zbiorem kluczy jest zbiór  $\mathcal{K} = \mathbb{Z}_m$  dla pewnej liczby całkowitej  $m$ .

Każdej linii  $W$  w obwodzie  $\mathbf{C}$  algorytm przypisuje wartość  $f(W) \in \mathcal{K}$ . Na początku linii wyjściowej  $W_{out}$  przypisuje się wartość klucza  $K$ , po czym algorytm iteruje się tyle razy, ile potrzeba, by każda linia miała przypisaną sobie wartość. Na koniec każdy uczestnik  $P_i$  dostaje listę wartości  $f(W)$ , takich że  $W$  jest linią wejściową obwodu, która są przesyłane dane wejściowe  $x_i$ .

Opis konstrukcji znajduje się na rysunku 11.3. Zauważmy, że gdy  $G$  jest bramką AND o, powiedzmy,  $t$  liniach wejściowych, rozdzielimy „klucz”  $f(W_G)$  między linie wejściowe, stosując schemat progowy o parametrach  $t, t$ .

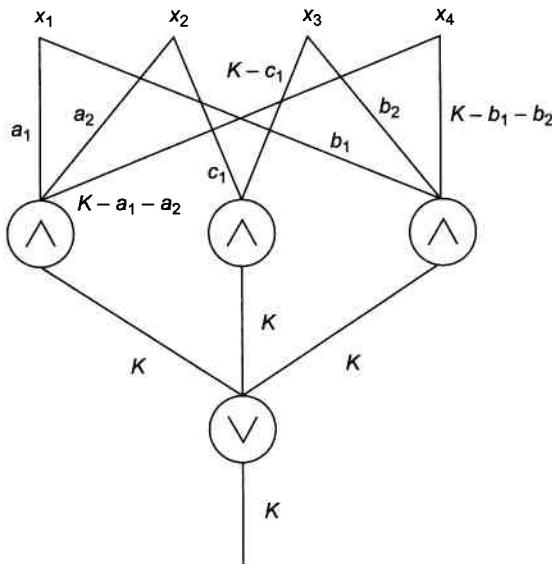
Wykonajmy opisaną wyżej procedurę dla schematu dostępu z przykładu 11.2, korzystając z obwodu odpowiadającego formule boolowskiej (11.1).

### Przykład 11.3

Rysunek 11.4 ilustruje konstrukcję, którą tu przeprowadzimy. Założmy, że kluczem jest  $K$ . Wartość  $K$  przypisujemy każdej z trzech linii wejściowych ostatniej bramki OR. Następnie rozpatrujemy bramkę AND odpowiadającą warunkowi

- (1)  $f(W_{out}) = K$
- (2) while istnieje linia, dla której  $f(W)$  nie jest określone do
- (3)     znajdź w C bramkę  $G$ , taką że  $f(W_G)$  jest określone dla linii wyjściowej  $W_G$  bramki  $G$ , ale  $f(W)$  nie jest określone dla żadnej linii wejściowej  $G$
- (4)     if  $G$  jest bramką OR then
- (5)          $f(W) = f(W_G)$  dla każdej linii wejściowej  $W$   
bramki  $G$
- (6)     else ( $G$  jest bramką AND)
- (7)         niech liniami wejściowymi  $G$  będą  $W_1, \dots, W_t$
- (8)         wybierz (losowo i niezależnie)  $t - 1$  elementów  
 $\mathbb{Z}_m: y_{G,1}, \dots, y_{G,t-1}$
- (9)         oblicz
- (10)          $y_{G,t} = f(W_G) - \sum_{i=1}^{t-1} y_{G,i} \bmod m$
- (11)         for  $1 \leq i \leq t$  do
- (11)              $f(W_i) = y_{G,i}$

RYSUNEK 11.3. Konstrukcja oparta na obwodzie monotonicznym



RYSUNEK 11.4. Obwód monotoniczny

$P_1 \wedge P_2 \wedge P_4$ . Jej trzem liniom wejściowym przypisujemy wartości  $a_1$ ,  $a_2$  oraz  $K - a_1 - a_2$ , odpowiednio, gdzie działania wykonujemy w  $\mathbb{Z}_m$ . Podobnie trzem liniom wejściowym odpowiadającym formule  $P_1 \wedge P_3 \wedge P_4$  przypisujemy wartości  $b_1$ ,  $b_2$  oraz  $K - b_1 - b_2$  i na koniec przypisujemy wartości  $c_1$  i  $K - c_1$  dwóm

liniom wejściowym odpowiadającym formule  $P_2 \wedge P_3$ . Przypomnijmy, że  $a_1, a_2, b_1, b_2$  i  $c_1$  są niezależnymi wartościami losowymi w  $\mathbb{Z}_m$ . Popatrzmy na udziały przyznane każdemu z czterech uczestników:

- (1)  $P_1$  dostaje  $a_1, b_1$ .
- (2)  $P_2$  dostaje  $a_2, c_1$ .
- (3)  $P_3$  dostaje  $b_2, K - c_1$ .
- (4)  $P_4$  dostaje  $K - a_1 - a_2, K - b_1 - b_2$ .

Tak więc udział każdego uczestnika składa się z dwóch elementów  $\mathbb{Z}_m$ .

Wykażemy, że schemat jest doskonały. Po pierwsze, sprawdzamy, że każdy zbiór z bazy może obliczyć klucz  $K$ . Upoważniony podzbiór  $\{P_1, P_2, P_4\}$  może obliczyć

$$K = a_1 + a_2 + (K - a_1 - a_2) \bmod m.$$

Dla podzbioru  $\{P_1, P_3, P_4\}$  wygląda to następująco:

$$K = b_1 + b_2 + (K - b_1 - b_2) \bmod m.$$

Wreszcie podzbiór  $\{P_2, P_3\}$  może obliczyć

$$K = c_1 + (K - c_1) \bmod m.$$

Widać, że każdy upoważniony podzbiór użytkowników potrafi obliczyć klucz  $K$ , możemy więc zająć się podzbiorami nieupoważnionymi. Zauważmy, że nie musimy interesować się wszystkimi takimi podzbiorami. Istotnie, jeśli  $B_1$  i  $B_2$  są podzbiorami nieupoważnionymi,  $B_1 \subseteq B_2$  i  $B_2$  nie jest w stanie obliczyć  $K$ , to  $B_1$  również nie może tego zrobić. Powiemy, że (nieupoważniony) podzbiór  $B \subseteq \mathcal{P}$  jest *maksymalnym* podzbiorem nieupoważnionym, jeśli każdy podzbiór  $B_1$ , taki że  $B_1 \supseteq B$  i  $B_1 \neq B$ , należy do  $\Gamma$ . Z powyższych rozważań wynika więc, że wystarczy sprawdzić, iż żaden maksymalny podzbiór nieupoważniony nie jest w stanie uzyskać żadnej wiedzy o kluczu  $K$ . W rozważanym przykładzie takimi maksymalnymi podzbiorami są:

$$\{P_1, P_2\}, \{P_1, P_3\}, \{P_1, P_4\}, \{P_2, P_4\}, \{P_3, P_4\}.$$

W każdym przypadku łatwo stwierdzić, że klucza  $K$  nie można obliczyć – ponieważ albo brakuje jakiegoś fragmentu „losowej” informacji, albo wszystkie udziały uczestników z danego podzbioru są losowe. Na przykład, podzbiór  $\{P_1, P_2\}$  dysponuje tylko losowymi wartościami  $a_1, b_1, a_2, c_1$ . Inny przykład: podzbiór  $\{P_3, P_4\}$  ma jedynie udziały  $b_2, K - c_1, K - a_1 - a_2, K - b_1 - b_2$ . Klucza  $K$  nie można obliczyć, gdyż  $c_1, a_1, a_2$  i  $b_1$  są nieznanymi wartościami losowymi. Żaden nieupoważniony podzbiór nie jest zatem w stanie ustalić klucza  $K$ .  $\square$

Jeśli do tej samej struktury dostępu zastosujemy inny obwód, otrzymamy inny schemat. Zobaczmy to na przykładzie struktury dostępu z przykładu 11.2.

### Przykład 11.4

Przekształćmy formułę (11.1) do tzw. koniunktywnej postaci normalnej:

$$(P_1 \vee P_2) \wedge (P_1 \vee P_3) \wedge (P_2 \vee P_3) \wedge (P_2 \vee P_4) \wedge (P_3 \vee P_4). \quad (11.2)$$

(Sprawdzenie tego, że jest to formula równoważna formule (11.1), pozostawiamy Czytelnikowi). Wynik zastosowania schematu z użyciem obwodu formuły (11.2) będzie następujący:

- (1)  $P_1$  dostaje  $a_1, a_2$ .
- (2)  $P_2$  dostaje  $a_1, a_3, a_4$ .
- (3)  $P_3$  dostaje  $a_2, a_3, K - a_1 - a_2 - a_3 - a_4$ .
- (4)  $P_4$  dostaje  $a_4, K - a_1 - a_2 - a_3 - a_4$ .

Szczegóły pozostawiamy Czytelnikowi do sprawdzenia. □

Udowodnimy teraz, że konstrukcja oparta na obwodzie monotonicznym zawsze prowadzi do doskonałego tajnego schematu współużytkowania.

### TWIERDZENIE 11.1

Niech  $\mathbf{C}$  będzie monotonicznym obwodem boolowskim. Wówczas konstrukcja oparta na  $\mathbf{C}$  daje w wyniku doskonały schemat współużytkowania realizujący strukturę dostępu  $\Gamma(\mathbf{C})$ .

**DOWÓD** Przeprowadzimy dowód indukcyjny ze względu na liczbę bramek w obwodzie  $\mathbf{C}$ . Jeśli  $\mathbf{C}$  składa się tylko z jednej bramki, to wynik jest oczywisty. Gdy tą jedyną bramką jest OR, każdy z uczestników dostaje do ręki klucz, a schemat realizuje strukturę dostępu złożoną ze wszystkich niepustych zbiorów uczestników. Z kolei gdy jedną bramką w obwodzie  $\mathbf{C}$  jest bramka AND z  $t$  wejściami, schemat staje się schematem progowym o parametrach  $t, t$ , przedstawionym na rysunku 11.2.

Dalej, przyjmijmy założenie indukcyjne: dla pewnej liczby całkowitej  $j > 1$  wszystkie obwody  $\mathbf{C}$  z mniej niż  $j$  bramkami dają w wyniku konstrukcji schemat realizujący  $\Gamma(\mathbf{C})$ . Niech teraz  $\mathbf{C}$  będzie obwodem z  $j$  bramkami. Rozpatrzmy „ostatnią” bramkę  $G$ . Może to być zarówno bramka OR, jak i bramka AND. Założymy najpierw, że  $G$  jest bramką OR i oznaczmy kable wejściowe tej bramki przez  $W_i$ ,  $1 \leq i \leq t$ . Każda z tych  $t$  linii jest linią wyjściową pewnego podobwodu obwodu  $\mathbf{C}$ ; niech  $\mathbf{C}_i$  oznacza podobwód prowadzący do linii  $W_i$ ,  $1 \leq i \leq t$ . Na mocy założenia indukcyjnego każdy z obwodów  $\mathbf{C}_i$  realizuje strukturę dostępu  $\Gamma_{\mathbf{C}_i}$ . Łatwo teraz zauważyc, że

$$\Gamma(\mathbf{C}) = \bigcup_{i=1}^t \Gamma_{\mathbf{C}_i}.$$

Ponieważ każdej linii  $W_i$  jest przypisana wartość klucza  $K$ , schemat rzeczywiście realizuje  $\Gamma(\mathbf{C})$ , zgodnie z tezą. Podobnie przebiega analiza, gdy  $G$  jest bramką AND. W tym przypadku

$$\Gamma(\mathbf{C}) = \bigcap_{i=1}^t \Gamma_{\mathbf{C}_i}.$$

Teza, że schemat realizuje  $\Gamma(\mathbf{C})$ , wynika teraz z tego, iż klucz  $K$  jest dzielony między  $t$  linii  $W_i$  za pomocą schematu progowego z parametrami  $t, t$ . To kończy dowód. ■

Oczywiście, gdy uczestnicy z upoważnionego podzbioru  $B$  chcą obliczyć klucz, muszą znać obwód użyty przez  $D$  do rozdzielenia udziałów, muszą też wieźć, jakie udziały odpowiadają poszczególnym liniom. Cała ta informacja ma charakter publiczny i tylko konkretne wartości udziałów pozostają tajne. Algorytm odtwarzania klucza wymagałączenia udziałów zgodnie z budową obwodu, przy czym bramka AND odpowiada dodawaniu wartości na liniach wejściowych modulo  $m$  (jeśli te wartości są znane), podczas gdy bramka OR odpowiada wyborowi wartości na dowolnej linii wejściowej (przy założeniu, że wszystkie te wartości są jednakowe).

#### 11.4. Formalne definicje

W tym podrozdziale podamy formalną definicję matematyczną (doskonałego) tajnego schematu współużytkowania. Taki schemat będzie reprezentowany przez zbiór reguł dystrybucji. *Reguła dystrybucji* jest dowolna funkcja

$$f : \mathcal{P} \rightarrow \mathcal{S}.$$

Reguła dystrybucji określa podział udziałów między uczestników, przy czym  $f(P_i)$  oznacza udział przypisany uczestnikowi  $P_i$ ,  $1 \leq i \leq w$ .

Dla każdego  $K \in \mathcal{K}$  niech  $\mathcal{F}_K$  będzie zbiorem reguł dystrybucji odpowiadających kluczowi o wartości  $K$ . Zbiory  $\mathcal{F}_K$  są publicznie znane.

Dalej, niech

$$\mathcal{F} = \bigcup_{K \in \mathcal{K}} \mathcal{F}_K.$$

$\mathcal{F}$  jest pełnym zbiorem reguł dystrybucji danego schematu. Jeśli  $K \in \mathcal{K}$  jest wartością klucza, który rozdający  $D$  chce rozdzielić między uczestników, to  $D$  wybiera regułę dystrybucji  $f \in \mathcal{F}_K$  i zgodnie z nią rozdaje udziały.

Tak wygląda najogólniejszy model, w którym będziemy badać tajne schematy współużytkowania. Każdy ze znanych nam schematów można opisać w tym kontekście, podając reguły dystrybucji, które są w nim stosowane. Matematyczna scisłość modelu ułatwia formułowanie definicji i konstrukcję dowodów.

Warto określić warunki, które zapewnią, że zbiór reguł dystrybucji schematu realizuje żądaną strukturę dostępu. Będzie to wymagało przyjrzenia się pewnym rozkładom prawdopodobieństwa, podobnie jak wtedy, gdy zajmowaliśmy się doskonałą tajnością. Na początek założymy więc, że mamy rozkład prawdopodobieństwa  $p_K$  na  $\mathcal{K}$ . Dla każdego  $K \in \mathcal{K}$  rozdający  $D$  wybierze ze zbioru  $\mathcal{F}_K$  regułę dystrybucji zgodnie z rozkładem prawdopodobieństwa  $p_{\mathcal{F}_K}$ .

Mając te rozkłady, nietrudno obliczyć rozkład prawdopodobieństwa dla każdej listy udziałów przydzielonych dowolnemu podzbiorowi uczestników  $B$  (upoważnionemu lub nie). Można to uczynić w sposób następujący. Niech  $B \subseteq \mathcal{P}$ . Określmy

$$\mathcal{S}(B) = \{f|_B : f \in \mathcal{F}\},$$

gdzie  $f|_B$  oznacza ograniczenie reguły dystrybucji  $f$  do zbioru  $B$ . Inaczej mówiąc,  $f|_B : B \rightarrow \mathcal{S}$  jest funkcją, taką że

$$f|_B(P_i) = f(P_i)$$

dla każdego  $P_i \in B$ . Tak więc  $\mathcal{S}(B)$  jest zbiorem wszystkich możliwych dystrybucji udziałów wśród uczestników ze zbioru  $B$ .

Obliczymy teraz rozkład prawdopodobieństwa na  $\mathcal{S}(B)$ , oznaczany symbolem  $p_{\mathcal{S}(B)}$ . Niech  $f_B \in \mathcal{S}(B)$ . Wówczas

$$p_{\mathcal{S}(B)}(f_B) = \sum_{K \in \mathcal{K}} p_K(K) \sum_{\{f \in \mathcal{F}_K : f|_B = f_B\}} p_{\mathcal{F}_K}(f).$$

Ponadto,

$$p_{\mathcal{S}(B)}(f_B|K) = \sum_{\{f \in \mathcal{F}_K : f|_B = f_B\}} p_{\mathcal{F}_K}(f),$$

dla wszystkich  $f_B \in \mathcal{S}(B)$  i  $K \in \mathcal{K}$ .

A oto formalna definicja doskonałego tajnego schematu współużytkowania.

### DEFINICJA 11.3

Niech  $\Gamma$  będzie strukturą dostępu, a  $\mathcal{F} = \bigcup_{K \in \mathcal{K}} \mathcal{F}_K$  zbiorem reguł dystrybucji. Wówczas  $\mathcal{F}$  jest *doskonalem tajnym schematem współużytkowania* realizującym strukturę dostępu  $\Gamma$ , jeśli spełnione są następujące dwa warunki:

- (1) Dla dowolnego upoważnionego podzbioru uczestników  $B \subseteq \mathcal{P}$  oraz reguły dystrybucji  $f \in \mathcal{F}_K$  i  $f' \in \mathcal{F}_{K'}$ , jeśli  $K \neq K'$ , to  $f|_B \neq f'|_B$  (a więc każda dystrybucja udziałów wśród uczestników z upoważnionego podzbioru  $B$  pozwala wyznaczyć wartość klucza).
- (2) Dla dowolnego nieupoważnionego podzbioru  $B \subseteq \mathcal{P}$  i dystrybucji  $f_B \in \mathcal{S}_B$ ,  $p_K(K|f_B) = p_K(K)$  dla każdego klucza  $K \in \mathcal{K}$  (czyli dla jakiegokolwiek dystrybucji  $f_B$  udziałów wśród uczestników z nieupoważnionego

podzbioru  $B$ , rozkład prawdopodobieństwa warunkowego na  $\mathcal{K}$  jest taki sam, jak aprioryczny rozkład prawdopodobieństwa na  $\mathcal{K}$ ; oznacza to, że udziały dostarczone uczestnikom z  $B$  nie zawierają żadnej informacji o wartości klucza).

Zauważmy, że drugi warunek definicji 11.3 jest bardzo podobny do pojęcia doskonałej tajności. To tłumaczy, dlaczego otrzymany tajny schemat współużytkowania określa się terminem „doskonały”.

Odrobjmy jeszcze i to, że prawdopodobieństwo  $p_{\mathcal{K}}(K|f_B)$  można uzyskać z opisanych wyżej rozkładów prawdopodobieństwa za pośrednictwem wzoru Bayesa:

$$p_{\mathcal{K}}(K|f_B) = \frac{p_{S_B}(f_B|K)p_{\mathcal{K}}(K)}{p_{S(B)}(f_B)}.$$

Zobaczmy przykład ilustrujący wprowadzone tu definicje.

### Przykład 11.5

Przedstawimy reguły dystrybucji dla schematu zbudowanego w przykładzie 11.4, gdy realizujemy go w  $\mathbb{Z}_2$ . Każdy ze zbiorów  $\mathcal{F}_0$  i  $\mathcal{F}_1$  zawiera 16 równoprawdopodobnych reguł dystrybucji. Dla uproszczenia zapisu binarny ciąg  $k$ -wyrazowy zastąpimy liczbą całkowitą zawartą między 0 i  $2^k - 1$ . Wówczas zbiory  $\mathcal{F}_0$  i  $\mathcal{F}_1$  możemy przedstawić tak jak na rysunku 11.5, na którym każdy wiersz reprezentuje regułę dystrybucji.

Dla dowolnego rozkładu prawdopodobieństwa na zbiorze kluczy  $\mathcal{K}$  otrzymujemy schemat doskonały. Nie będziemy tu wykonywać wszystkich obliczeń, zatrzymamy się jedynie na kilku typowych przypadkach ilustrujących rolę warunków, o których mówi definicja 11.3.

Zbiór  $\{P_2, P_3\}$  jest podzbiorem upoważnionym, a więc udziały uczestników  $P_2$  i  $P_3$  powinny (łącznie) umożliwić ustalenie wartości klucza. Łatwo sprawdzić, że każdy przydział udziałów dla tych dwóch uczestników występuje w regule dystrybucji należącej do co najwyżej jednego ze zbiorów  $\mathcal{F}_0$  i  $\mathcal{F}_1$ . Na przykład, jeśli udziałem  $P_2$  jest 3, a  $P_3$  dostał 6, to reguła musi być ósma reguła dystrybucji w  $\mathcal{F}_0$  i w rezultacie wartością klucza jest 0.

Jednak  $B = \{P_1, P_2\}$  jest podzbiorem nieupoważnionym. Nietrudno sprawdzić, że każdy przydział udziałów dla tych dwóch uczestników występuje w dokładnie jednej regule dystrybucji ze zbioru  $\mathcal{F}_0$  oraz w dokładnie jednej regule dystrybucji ze zbioru  $\mathcal{F}_1$ . Stąd

$$p_{S(B)}(f_B|K) = \frac{1}{16}$$

| $\mathcal{F}_0$ |       |       |       | $\mathcal{F}_1$ |       |       |       |
|-----------------|-------|-------|-------|-----------------|-------|-------|-------|
| $P_1$           | $P_2$ | $P_3$ | $P_4$ | $P_1$           | $P_2$ | $P_3$ | $P_4$ |
| 0               | 0     | 0     | 0     | 0               | 0     | 1     | 1     |
| 0               | 1     | 1     | 3     | 0               | 1     | 0     | 2     |
| 0               | 2     | 3     | 1     | 0               | 2     | 2     | 0     |
| 0               | 3     | 2     | 2     | 0               | 3     | 3     | 3     |
| 1               | 0     | 4     | 0     | 1               | 0     | 5     | 1     |
| 1               | 1     | 5     | 3     | 1               | 1     | 4     | 2     |
| 1               | 2     | 7     | 1     | 1               | 2     | 6     | 0     |
| 1               | 3     | 6     | 2     | 1               | 3     | 7     | 3     |
| 2               | 4     | 0     | 0     | 2               | 4     | 1     | 1     |
| 2               | 5     | 1     | 3     | 2               | 5     | 0     | 0     |
| 2               | 6     | 3     | 1     | 2               | 6     | 2     | 2     |
| 2               | 7     | 2     | 2     | 2               | 7     | 3     | 3     |
| 3               | 4     | 4     | 0     | 3               | 4     | 5     | 1     |
| 3               | 5     | 5     | 3     | 3               | 5     | 4     | 2     |
| 3               | 6     | 7     | 1     | 3               | 6     | 6     | 0     |
| 3               | 7     | 6     | 2     | 3               | 7     | 7     | 3     |

RYSUNEK 11.5. Reguły dystrybucji dla tajnego schematu współużytkowania

dla każdej funkcji  $f_B \in \mathcal{S}(B)$  i  $K = 0, 1$ . Teraz obliczamy

$$\begin{aligned} p_{\mathcal{S}(B)}(f_B) &= \sum_{K \in \mathcal{K}} p_{\mathcal{K}}(K) \sum_{\{f \in \mathcal{F}_K : f|_B = f_B\}} p_{\mathcal{F}_K}(f) \\ &= \sum_{K=0}^1 p_{\mathcal{K}}(K) \cdot \frac{1}{16} \\ &= \frac{1}{16}. \end{aligned}$$

Dalej korzystamy ze wzoru Bayesa, by obliczyć  $p_{\mathcal{K}}(K|f_B)$ :

$$\begin{aligned} p_{\mathcal{K}}(K|f_B) &= \frac{p_{\mathcal{S}_B}(f_B|K)p_{\mathcal{K}}(K)}{p_{\mathcal{S}(B)}(f_B)} \\ &= \frac{\frac{1}{16} \cdot p_{\mathcal{K}}(K)}{\frac{1}{16}} \\ &= p_{\mathcal{K}}(K) \end{aligned}$$

Tak więc drugi warunek jest spełniony dla tego podzbioru  $B$ .

Podobne obliczenia można wykonać dla pozostałych podzbiorów, upoważnionych i nieupoważnionych, by stwierdzić, że w każdym przypadku spełniony jest odpowiedni warunek. Mamy zatem doskonały tajny schemat współużytkowania.  $\square$

### 11.5. Względna miara informacji

Wyniki uzyskane w podrozdziale 11.3 dowodzą, że każdą monotoniczną strukturę dostępu można zrealizować w doskonałym tajnym schemacie współużytkowania. Zajmiemy się teraz efektywnością otrzymanych schematów. W przypadku schematu progowego o parametrach  $t, w$  możemy zbudować obwód odpowiadający dysjunktywnej postaci normalnej formuły boolowskiej, który będzie miał  $1 + \binom{w}{t}$  bramek. Udział każdego uczestnika składałby się z  $\binom{w-1}{t-1}$  elementów  $\mathbb{Z}_m$ . Takie rozwiązanie wydaje się bardzo nieefektywne, gdyż schemat progowy Shama o parametrach  $t, w$  umożliwia dzielenie klucza przez przypisanie każdemu uczestnikowi jednego „elementu” informacji.

Ogólniej, wydajność tajnego schematu współużytkowania mierzymy względną miarą informacji, którą teraz zdefiniujemy.

#### DEFINICJA 11.4

Niech dany będzie doskonały tajny schemat współużytkowania realizujący strukturę dostępu  $\Gamma$ . *Względną miarę informacji* uczestnika  $P_i$  nazywamy wielkość

$$\rho_i = \frac{\log_2 |\mathcal{K}|}{\log_2 |\mathcal{S}(P_i)|}.$$

(Zauważmy, że  $\mathcal{S}(P_i)$  oznacza zbiór wszystkich możliwych udziałów, które mogą przypaść uczestnikowi  $P_i$ ; oczywiście  $\mathcal{S}(P_i) \subseteq \mathcal{S}$ ). *Względną miarę informacji schematu* oznaczamy symbolem  $\rho$  i określamy w sposób następujący:

$$\rho = \min\{\rho_i : 1 \leq i \leq w\}.$$

Jakie jest uzasadnienie tej definicji? Klucz  $K$  pochodzi ze skończonego zbioru  $\mathcal{K}$ , możemy więc utożsamić  $K$  z ciągiem bitów o długości  $\log_2 |\mathcal{K}|$ , na przykład, kodując go w systemie dwójkowym. Podobnie udział przydzielony uczestnikowi  $P_i$  można zapisać w postaci ciągu  $\log_2 |\mathcal{S}(P_i)|$  bitów. Intuicyjnie,  $P_i$  otrzymuje (jako swój udział)  $\log_2 |\mathcal{S}(P_i)|$  bitów informacji, podczas gdy klucz zawiera  $\log_2 |\mathcal{K}|$  bitów; tak więc  $\rho_i$  określa stosunek liczby bitów w jednym udziale do liczby bitów klucza.

#### Przykład 11.6

Popatrzymy znów na oba schematy z podrozdziału 11.2. Dla schematu z przykładu 11.3 mamy

$$\rho = \frac{\log_2 m}{\log_2 m^2} = \frac{1}{2}.$$

Z kolei w przykładzie 11.4 otrzymujemy schemat, dla którego

$$\rho = \frac{\log_2 m}{\log_2 m^3} = \frac{1}{3}.$$

Pierwsza z tych implementacji okazuje się zatem lepsza. □

Na ogólnie, gdy budujemy schemat na podstawie obwodu **C** (metodą konstrukcji opartej na obwodzie monotonicznym), względną miarę informacji można obliczyć w sposób opisany w następującym twierdzeniu.

### TWIERDZENIE 11.2

Niech **C** będzie dowolnym monotonicznym obwodem boolowskim. Wówczas istnieje doskonały tajny schemat współużytkowania realizujący strukturę dostępu  $\Gamma(\mathbf{C})$ , którego względna miara informacji jest równa

$$\rho = \max\{1/r_i : 1 \leq i \leq w\},$$

gdzie  $r_i$  oznacza liczbę linii wejściowych obwodu **C** z wejściem  $x_i$ .

W przypadku progowych struktur dostępu widać, że względna miara informacji w schemacie Shamira jest równa 1, co, jak wykażemy dalej, jest wartością optymalną. Jednakże implementacja schematu progowego o parametrach  $t, w$  z użyciem dysjunktywnej postaci normalnej obwodu boolowskiego daje względna miarę informacji równą  $1/(w-1)$ , a więc znacznie mniejszą (i przez to gorszą), gdy  $1 < t < w$ .

Pożądana jest oczywiście jak największa względna miara informacji. Pierwszy ogólny wynik, przedstawiony niżej, stwierdza, że w dowolnym schemacie mamy  $\rho \leq 1$ .

### TWIERDZENIE 11.3

W każdym doskonałym tajnym schemacie współużytkowania realizującym strukturę dostępu  $\Gamma$  jest  $\rho \leq 1$ .

**DOWÓD** Założymy, że mamy doskonały tajny schemat współużytkowania realizujący strukturę dostępu  $\Gamma$ . Niech  $B \in \Gamma_0$  i niech  $P_j$  będzie dowolnym uczestnikiem ze zbioru  $B$ . Określmy zbiór  $B' = B \setminus \{P_j\}$ ; wówczas  $B' \notin \Gamma$ . Stąd, jeśli  $g \in \mathcal{S}(B)$ , to dystrybucja udziałów  $g|_{B'}$  nie dostarcza żadnej informacji o kluczu. W konsekwencji, dla każdego  $K \in \mathcal{K}$  istnieje reguła dystrybucji  $g^K \in \mathcal{F}_K$ , taka że  $g^K|_{B'} = g|_{B'}$ . Ponieważ  $B \in \Gamma$ , zatem musi być tak, że  $g^K(P_j) \neq g^{K'}(P_j)$ , gdy  $K \neq K'$ . Okazuje się więc, że  $|\mathcal{S}(P_j)| \geq |\mathcal{K}|$ , a więc  $\rho \leq 1$ . ■

Jak widać, mamy optymalną sytuację wtedy, gdy  $\rho = 1$ . Schemat o tej właściwości nazwiemy schematem *idealnym*. I tak, schematami idealnymi są schematy Shamira. W następnym podrozdziale opiszemy konstrukcję schematów idealnych uogólniającą schematy Shamira.

## 11.6. Konstrukcja Brickella oparta na przestrzeni liniowej

W tym podrozdziale opiszemy konstrukcję schematów idealnych znaną jako *konstrukcja Brickella oparta na przestrzeni liniowej*.

Niech  $\Gamma$  będzie strukturą dostępu i niech  $(\mathbb{Z}_p)^d$  będzie przestrzenią liniową wszystkich  $d$ -wyrazowych ciągów elementów  $\mathbb{Z}_p$ , gdzie  $p$  jest liczbą pierwszą i  $d \geq 2$ . Założymy dalej, że istnieje funkcja

$$\phi : \mathcal{P} \rightarrow (\mathbb{Z}_p)^d$$

o następujących własnościach

$$(1, 0, \dots, 0) \in \langle \phi(P_i) : P_i \in B \rangle \Leftrightarrow B \in \Gamma. \quad (11.3)$$

Innymi słowy, wektor  $(1, 0, \dots, 0)$  można przedstawić w postaci kombinacji liniowej wektorów ze zbioru  $\{\phi(P_i) : P_i \in B\}$  wtedy i tylko wtedy, gdy  $B$  jest podzbiorem upoważnionym.

Założymy teraz, że istnieje funkcja  $\phi$  spełniająca warunek (11.3). (Na ogólną taką funkcję znajduje się metodą prób i błędów, chociaż poznamy dalej pewne konstrukcje, które dają w wyniku tego rodzaju funkcje dla pewnych struktur dostępu).

Zbudujemy idealny tajny schemat współużytkowania, dla którego  $\mathcal{K} = \mathcal{S}(P_i) = \mathbb{Z}_p$ ,  $1 \leq i \leq w$ . Reguły dystrybucji w tym schemacie są następujące: dla każdego wektora  $\bar{a} = (a_1, \dots, a_d) \in (\mathbb{Z}_p)^d$  określamy regułę dystrybucji  $f_{\bar{a}} \in \mathcal{F}_{a_1}$ , taką że

$$f_{\bar{a}}(x) = \bar{a} \cdot \phi(x)$$

dla każdego  $x \in \mathcal{P}$ ; „ $\cdot$ ” oznacza tu iloczyn skalarny modulo  $p$ .

### Faza inicjowania

- (1)  $D$  przekazuje wektor  $\phi(P_i) \in (\mathbb{Z}_p)^d$  do  $P_i$ , dla  $1 \leq i \leq w$ ; wektory są znane publicznie

### Rozdanie udziałów

- (2) założymy, że  $D$  chce rozdać klucz  $K \in \mathbb{Z}_p$ ; wybiera potajemnie (lokalnie i niezależnie)  $d - 1$  elementów  $\mathbb{Z}_p$ :  $a_2, \dots, a_d$

- (3)  $D$  oblicza  $y_i = \bar{a} \cdot \phi(P_i)$  dla  $1 \leq i \leq w$ , gdzie

$$\bar{a} = (K, a_2, \dots, a_d)$$

- (4)  $D$  przekazuje udział  $y_i$  do  $P_i$ , dla  $1 \leq i \leq w$

### RYSUNEK 11.6. Schemat Brickella

Zauważmy, że każdy ze zbiorów  $\mathcal{F}_K$  zawiera  $p^{d-1}$  reguły dystrybucji. Przyjmijmy, że każda reguła jest równie prawdopodobna, czyli  $p_{\mathcal{F}_K}(f) = 1/p^{d-1}$  dla każdego  $f \in \mathcal{F}_K$ . Schemat Brickella przedstawiamy na rysunku 11.6.

Mamy następujący wynik.

**TWIERDZENIE 11.4**

Niech  $\phi$  będzie funkcją spełniającą warunek (11.3). Wówczas zbiory reguł dystrybucji  $\mathcal{F}_K$ ,  $K \in \mathcal{K}$ , stanowią idealny schemat realizujący  $\Gamma$ .

**DOWÓD** Wykażemy najpierw, że jeśli  $B$  jest podzbiorem upoważnionym, to uczestnicy należący do  $B$  mogą obliczyć klucz  $K$ . Z założenia,

$$(1, 0, \dots, 0) \in \langle \phi(P_i) : P_i \in B \rangle,$$

gdzie  $c_i \in \mathbb{Z}_p$  dla każdego  $i$ ,

$$(1, 0, \dots, 0) = \sum_{\{i: P_i \in B\}} c_i \phi(P_i).$$

Niech  $s_i$  oznacza udział przypisany uczestnikowi  $P_i$ . Wtedy

$$s_i = \bar{a} \cdot \phi(P_i),$$

gdzie  $\bar{a}$  jest nieznanym wektorem wybranym przez  $D$  oraz

$$K = a_1 = \bar{a} \cdot (1, 0, \dots, 0).$$

Z liniowości iloczynu skalarnego otrzymujemy

$$K = \sum_{\{i: P_i \in B\}} c_i \bar{a} \cdot \phi(P_i).$$

Tak więc uczestnicy ze zbioru  $B$  mogą bez trudu obliczyć

$$K = \sum_{\{i: P_i \in B\}} c_i s_i.$$

Co się dzieje, gdy  $B$  nie jest upoważnionym podzbiorem? Niech  $e$  oznacza wymiar podprzestrzeni  $\langle \phi(P_i) : P_i \in B \rangle$  (zwróćmy uwagę na to, że  $e \leq |B|$ ). Wybierzmy dowolny element  $K \in \mathcal{K}$  i rozważmy układ równań:

$$\phi(P_i) \cdot \bar{a} = s_i, \quad \forall P_i \in B$$

$$(1, 0, \dots, 0) \cdot \bar{a} = K.$$

Jest to układ równań liniowych z  $d$  niewiadomymi  $a_1, \dots, a_d$ . Macierz współczynników ma rzad  $e+1$ , ponieważ

$$(1, 0, \dots, 0) \notin \langle \phi(P_i) : P_i \in B \rangle.$$

Jeśli zatem układ jest niesprzeczny, przestrzeń rozwiązań ma wymiar  $d - e - 1$  (niezależnie od wartości  $K$ ). Wynika stąd, że istnieją dokładnie  $p^{d-e-1}$  reguły dystrybucji w każdym ze zbiorów  $\mathcal{F}_K$ , zgodne z dowolną dystrybucją udziałów wśród uczestników ze zbioru  $B$ . Drogą obliczeń podobnych do tych z przykładu 11.5 stwierdzamy, że  $p_K(K|f_B) = p_K(K)$  dla każdego  $K \in \mathcal{K}$ , gdzie  $f_B(P_i) = s_i$  dla wszystkich  $P_i \in B$ .

Pozostaje zatem wykazać, że układ równań jest niesprzeczny. Układ pierwotnych  $|B|$  równań jest taki, gdyż wybrany przez  $D$  wektor  $\bar{a}$  jest jego rozwiązaniem. Ponadto, jak już wiemy,

$$(1, 0, \dots, 0) \notin \langle \phi(P_i) : P_i \in B \rangle,$$

zatem ostatnie równanie jest niesprzeczne z pierwszymi  $|B|$  równaniami. To kończy dowód. ■

Warto odnotować, że schemat progowy Shamira o parametrach  $t, w$  jest szczególnym przypadkiem konstrukcji opartej na przestrzeni liniowej. Aby się o tym przekonać, przyjmijmy  $d = t$  oraz

$$\phi(P_i) = (1, x_i, x_i^2, \dots, x_i^{t-1})$$

dla  $1 \leq i \leq w$ , gdzie  $x_i$  jest pierwszą współrzędną punktu przydzielonego uczestnikowi  $P_i$ . Otrzymany w ten sposób schemat jest równoważny schematowi Shamira; szczegóły pozostawiamy do sprawdzenia Czytelnikowi.

A oto następny ogólny wynik, który jest łatwy do udowodnienia. Dotyczy on struktur dostępu, w których bazą jest zbiór par uczestników stanowiący pełny graf wielodzielny. Graf  $G = (V, E)$  ze zbiorami wierzchołków  $V$  i krawędzi  $E$  jest *pełnym grafem wielodzielnym*, jeśli zbiór  $V$  można rozbić na podzbiory  $V_1, \dots, V_\ell$ , tak że  $\{x, y\} \in E$  wtedy i tylko wtedy, gdy  $x \in V_i, y \in V_j$  dla  $1 \leq i, j \leq \ell$ , takich że  $i \neq j$ . Zbiory  $V_i$  nazwiemy *składowymi*. Gdy  $|V_i| = n_i$  dla  $1 \leq i \leq \ell$ , pełny graf wielodzielny oznaczamy symbolem  $K_{n_1, \dots, n_\ell}$ . Pełny graf wielodzielny  $K_{1, \dots, 1}$  ( $\ell$  składowymi) jest po prostu *grafem pełnym*, który oznaczamy symbolem  $K_\ell$ .

### TWIERDZENIE 11.5

Niech  $G = (V, E)$  będzie pełnym grafem wielodzielnym. Istnieje wówczas idealny schemat realizujący strukturę dostępu  $cl(E)$  na zbiorze uczestników  $V$ .

**DOWÓD** Niech  $V_1, \dots, V_\ell$  będą składowymi grafu  $G$  i niech  $x_1, \dots, x_\ell$  będą różnymi elementami  $\mathbb{Z}_p$ , przy czym  $p \geq \ell$ . Przyjmijmy  $d = 2$ . Dla każdego uczestnika  $v \in V_i$  określmy  $\phi(v) = (x_i, 1)$ . Bez trudu sprawdzamy, że spełniony jest warunek (11.3), zatem wnioskujemy z twierdzenia 11.4, że otrzymany schemat jest schematem idealnym. ■

Aby zilustrować przedstawione wyżej konstrukcje, rozważymy struktury dostępu obejmujące co najwyżej czterech uczestników. Zauważmy, że wystarczy rozpatrywać tylko takie struktury, w których bazy nie można rozbić na dwa niepuste podzbiory złożone z rozłącznych zbiorów uczestników. (Nie rozpatrujemy, na przykład, bazy  $\Gamma_0 = \{\{P_1, P_2\}, \{P_3, P_4\}\}$ , gdyż można ją przedstawić w postaci sumy  $\{\{P_1, P_2\}\} \cup \{\{P_3, P_4\}\}$ ). Wszystkie nieizomorficzne struktury tego typu z dwoma, trzema lub czterema uczestnikami przedstawiamy w tablicy 11.1 (wartość  $\rho^*$  zdefiniujemy w podrozdziale 11.7).

**TABLICA 11.1.** Struktury dostępu dla co najwyżej czterech uczestników

|    | $w$ | Podzbiory $\Gamma_0$                             | $\rho^*$ | Uwagi                      |
|----|-----|--------------------------------------------------|----------|----------------------------|
| 1  | 2   | $P_1P_2$                                         | 1        | próg (2,2)                 |
| 2  | 3   | $P_1P_2, P_2P_3$                                 | 1        | $\Gamma_0 \cong K_{1,2}$   |
| 3  | 3   | $P_1P_2, P_2P_3, P_1P_3$                         | 1        | próg (2,3)                 |
| 4  | 3   | $P_1P_2P_3$                                      | 1        | próg (3,3)                 |
| 5  | 4   | $P_1P_2, P_2P_3, P_3P_4$                         | 2/3      |                            |
| 6  | 4   | $P_1P_2, P_1P_3, P_1P_4$                         | 1        | $\Gamma_0 \cong K_{1,3}$   |
| 7  | 4   | $P_1P_2, P_1P_4, P_2P_3, P_3P_4$                 | 1        | $\Gamma_0 \cong K_{2,2}$   |
| 8  | 4   | $P_1P_2, P_2P_3, P_2P_4, P_3P_4$                 | 2/3      |                            |
| 9  | 4   | $P_1P_2, P_1P_3, P_1P_4, P_2P_3, P_2P_4$         | 1        | $\Gamma_0 \cong K_{1,1,2}$ |
| 10 | 4   | $P_1P_2, P_1P_3, P_1P_4, P_2P_3, P_2P_4, P_3P_4$ | 1        | próg (2,4)                 |
| 11 | 4   | $P_1P_2P_3, P_1P_4$                              | 1        |                            |
| 12 | 4   | $P_1P_3P_4, P_1P_2, P_2P_3$                      | 2/3      |                            |
| 13 | 4   | $P_1P_3P_4, P_1P_2, P_2P_3, P_2P_4$              | 2/3      |                            |
| 14 | 4   | $P_1P_2P_3, P_1P_2P_4$                           | 1        |                            |
| 15 | 4   | $P_1P_2P_3, P_1P_2P_4, P_3P_4$                   | 1        |                            |
| 16 | 4   | $P_1P_2P_3, P_1P_2P_4, P_1P_3P_4$                | 1        |                            |
| 17 | 4   | $P_1P_2P_3, P_1P_2P_4, P_1P_3P_4, P_2P_3P_4$     | 1        | próg (3,4)                 |
| 18 | 4   | $P_1P_2P_3P_4$                                   | 1        | próg (4,4)                 |

Wśród tych 18 struktur dostępu jest 10, z których możemy utworzyć idealne schematy za pomocą znanych nam już konstrukcji. Są to albo struktury progowe, albo takie, w których baza jest pełnym grafem wielodzielonym, a więc można do nich stosować twierdzenie 11.5. Jedną z nich jest struktura numer 9, której bazą jest pełny graf wielodzielny  $K_{1,1,2}$ . Zilustrujemy to za pomocą następującego przykładu.

### Przykład 11.7

Przyjmijmy dla struktury numer 9 wartości:  $d = 2$ ,  $p \geq 3$  i określmy funkcję  $\phi$  w następujący sposób:

$$\begin{aligned}\phi(P_1) &= (0, 1) \\ \phi(P_2) &= (1, 1) \\ \phi(P_3) &= (2, 1) \\ \phi(P_4) &= (2, 1)\end{aligned}$$

Z twierdzenia 11.5 wynika, że otrzymany schemat jest idealny. □

Pozostaje do rozważenia jeszcze osiem struktur dostępu. W przypadku czterech z nich o numerach: 11, 14, 15 i 16 można zastosować *ad hoc* konstrukcję opartą na przestrzeni liniowej do zbudowania schematu idealnego. Opiszemy tu konstrukcje dla struktur 11 i 14.

*Przykład 11.8*

Przyjmijmy dla struktury numer 11 wartości:  $d = 3$ ,  $p \geq 3$  i określmy funkcję  $\phi$  w następujący sposób:

$$\begin{aligned}\phi(P_1) &= (0, 1, 0) \\ \phi(P_2) &= (1, 0, 1) \\ \phi(P_3) &= (0, 1, -1) \\ \phi(P_4) &= (1, 1, 0)\end{aligned}$$

Zauważmy najpierw, że

$$\begin{aligned}\phi(P_4) - \phi(P_1) &= (1, 1, 0) - (0, 1, 0) \\ &= (1, 0, 0)\end{aligned}$$

oraz

$$\begin{aligned}\phi(P_2) + \phi(P_3) - \phi(P_1) &= (1, 0, 1) + (0, 1, -1) - (0, 1, 0) \\ &= (1, 0, 0).\end{aligned}$$

Stąd

$$(1, 0, 0) \in \langle \phi(P_1), \phi(P_2), \phi(P_3) \rangle$$

i

$$(1, 0, 0) \in \langle \phi(P_1), \phi(P_4) \rangle.$$

Wystarczy teraz wykazać, że

$$(1, 0, 0) \notin \langle \phi(P_i) : P_i \in B \rangle,$$

gdy  $B$  nie jest zbiorem upoważnionym. Mamy trzy takie podzbiory:  $\{P_1, P_2\}$ ,  $\{P_1, P_3\}$  oraz  $\{P_2, P_3, P_4\}$ . W każdym przypadku musimy stwierdzić, że odpowiedni układ równań liniowych nie ma rozwiązania. Na przykład, założymy, że

$$(1, 0, 0) = a_2\phi(P_2) + a_3\phi(P_3) + a_4\phi(P_4)$$

dla pewnych  $a_2, a_3, a_4 \in \mathbb{Z}_p$ . Otrzymujemy stąd równoważny układ równań:

$$\begin{aligned}a_2 + a_4 &= 1 \\ a_3 + a_4 &= 0 \\ a_2 - a_3 &= 0.\end{aligned}$$

Łatwo zauważyc, że układ ten nie ma rozwiązań. Pozostawiamy Czytelnikowi do sprawdzenia pozostałe dwa nieupoważnione podzbiory.  $\square$

### Przykład 11.9

Przyjmijmy dla struktury numer 14 wartości  $d = 3$ ,  $p \geq 2$  i określmy funkcję  $\phi$  w następujący sposób:

$$\phi(P_1) = (0, 1, 0)$$

$$\phi(P_2) = (1, 0, 1)$$

$$\phi(P_3) = (0, 1, 1)$$

$$\phi(P_4) = (0, 1, 1)$$

Znow stwierdzamy, że spełniony jest warunek (11.3), a więc wynikiem konstrukcji jest schemat idealny.  $\square$

Konstrukcje idealnych schematów dla struktur dostępu o numerach 15 i 16 pozostawiamy jako ćwiczenie. Zajmijmy się teraz wykazaniem, że nierożpatrzonych dotąd czterech struktur nie można zrealizować za pomocą schematu idealnego.

## 11.7. Górnne ograniczenie względnej miary informacji

Pozostały do rozważenia cztery struktury: 5, 8, 12 i 13. Dla żadnego z tych przykładów, jak się przekonamy, nie istnieje schemat ze względną miarą informacji  $\rho > 2/3$ .

Niech  $\rho^* = \rho^*(\Gamma)$  oznacza maksymalną względną miarę informacji doskonałego tajnego schematu współużytkowania realizującego daną strukturę dostępu  $\Gamma$ . Pierwszy wynik, który tu przedstawimy, wyznacza ograniczenie związane z entropią, z którego wyprowadzimy następnie górnne ograniczenie dotyczące  $\rho^*$  dla pewnych struktur dostępu. Określiliśmy rozkład prawdopodobieństwa  $p_K$  na  $\mathcal{K}$ ; niech  $H(\mathbf{K})$  będzie entropią tego rozkładu. Podobnie niech  $H(\mathbf{B})$  oznacza entropię rozkładu prawdopodobieństwa  $p_{S(B)}$  na udziałach przypisanych uczestnikom ze zbioru  $B \subseteq \mathcal{P}$ .

Zaczniemy od podania innej definicji doskonałego tajnego schematu współużytkowania, tym razem w języku entropii, równoważnej definicji 11.3.

### DEFINICJA 11.5

Niech  $\Gamma$  będzie strukturą dostępu, a  $\mathcal{F}$  zbiorem reguł dystrybucji. Wówczas  $\mathcal{F}$  jest doskonałym tajnym schematem współużytkowania realizującym strukturę  $\Gamma$ , jeśli spełnione są następujące dwa warunki:

- (1) Dla każdego upoważnionego zbioru uczestników  $B \subseteq \mathcal{P}$ ,  $H(\mathbf{K}|\mathbf{B}) = 0$ .
- (2) Dla każdego nieupoważnionego zbioru uczestników  $B \subseteq \mathcal{P}$ ,  $H(\mathbf{K}|\mathbf{B}) = H(\mathbf{K})$ .

Potrzebne nam będą różne równości i nierówności dotyczące entropii. Niektóre z nich poznaliśmy w podrozdziale 2.3; pozostałe dowodzi się podobnie, przytoczymy je więc tu bez dowodu.

### LEMAT 11.6

Niech  $\mathbf{X}$ ,  $\mathbf{Y}$  i  $\mathbf{Z}$  będą zmiennymi losowymi. Wówczas

$$H(\mathbf{XY}) = H(\mathbf{X}|\mathbf{Y}) + H(\mathbf{Y}) \quad (11.4)$$

$$H(\mathbf{XY}|\mathbf{Z}) = H(\mathbf{X}|\mathbf{YZ}) + H(\mathbf{Y}|\mathbf{Z}) \quad (11.5)$$

$$H(\mathbf{XY}|\mathbf{Z}) = H(\mathbf{Y}|\mathbf{XZ}) + H(\mathbf{X}|\mathbf{Z}) \quad (11.6)$$

$$H(\mathbf{X}|\mathbf{Y}) \geq 0 \quad (11.7)$$

$$H(\mathbf{X}|\mathbf{Z}) \geq H(\mathbf{X}|\mathbf{YZ}) \quad (11.8)$$

$$H(\mathbf{XY}|\mathbf{Z}) \geq H(\mathbf{Y}|\mathbf{Z}) \quad (11.9)$$

Udowodnimy teraz dwa lematy dotyczące entropii w tajnych schematach współużytkowania.

### LEMAT 11.7

Niech  $\Gamma$  będzie strukturą dostępu, a  $\mathcal{F}$  zbiorem reguł dystrybucji realizującym  $\Gamma$ . Jeśli  $A, B \subseteq \mathcal{P}$  oraz  $B \notin \Gamma$  i  $A \cup B \subseteq \mathcal{P}$ , to

$$H(\mathbf{A}|\mathbf{B}) = H(\mathbf{K}) + H(\mathbf{A}|\mathbf{BK}).$$

**DOWÓD** Z równań 11.5 i 11.6 otrzymujemy

$$H(\mathbf{AK}|\mathbf{B}) = H(\mathbf{A}|\mathbf{BK}) + H(\mathbf{K}|\mathbf{B})$$

oraz

$$H(\mathbf{AK}|\mathbf{B}) = H(\mathbf{K}|\mathbf{AB}) + H(\mathbf{A}|\mathbf{B}),$$

zatem

$$H(\mathbf{A}|\mathbf{BK}) + H(\mathbf{K}|\mathbf{B}) = H(\mathbf{K}|\mathbf{AB}) + H(\mathbf{A}|\mathbf{B}).$$

Z własności 2 w definicji 11.5 mamy

$$H(\mathbf{K}|\mathbf{B}) = H(\mathbf{K}),$$

a z własności 1 w tej definicji mamy

$$H(\mathbf{K}|\mathbf{AB}) = 0,$$

a stąd wynika już bezpośrednio teza lematu. ■

**LEMAT 11.8**

Niech  $\Gamma$  będzie strukturą dostępu, a  $\mathcal{F}$  zbiorem reguł dystrybucji realizującym  $\Gamma$ . Jeśli  $A, B \subseteq \mathcal{P}$  oraz  $A \cup B \notin \Gamma$ , to  $H(\mathbf{A}|\mathbf{B}) = H(\mathbf{A}|\mathbf{BK})$ .

**DOWÓD** Podobnie jak w lemacie 11.7, mamy

$$H(\mathbf{A}|\mathbf{BK}) + H(\mathbf{K}|\mathbf{B}) = H(\mathbf{K}|\mathbf{AB}) + H(\mathbf{A}|\mathbf{B}).$$

Teza wynika bezpośrednio z równości

$$H(\mathbf{K}|\mathbf{B}) = H(\mathbf{K})$$

oraz

$$H(\mathbf{K}|\mathbf{AB}) = H(\mathbf{K}).$$

|

Możemy teraz przystąpić do udowodnienia ważnego twierdzenia.

**TWIERDZENIE 11.9**

Niech  $\Gamma$  będzie strukturą dostępu, taką że

$$\{W, X\}, \{X, Y\}, \{W, Y, Z\} \in \Gamma$$

oraz

$$\{W, Y\}, \{X\}, \{W, Z\} \notin \Gamma.$$

Wówczas dla dowolnego doskonałego tajnego schematu współużytkowania  $\mathcal{F}$  realizującego  $\Gamma$  zachodzi  $H(\mathbf{XY}) \geq 3H(\mathbf{K})$ .

**DOWÓD** Mamy następujący ciąg nierówności:

$$\begin{aligned} H(\mathbf{K}) &= H(\mathbf{Y}|\mathbf{WZ}) - H(\mathbf{Y}|\mathbf{WZK}) && \text{z lematu 11.7} \\ &\leq H(\mathbf{Y}|\mathbf{WZ}) && \text{z (11.7)} \\ &\leq H(\mathbf{Y}|\mathbf{W}) && \text{z (11.8)} \\ &= H(\mathbf{Y}|\mathbf{WK}) && \text{z lematu 11.8} \\ &\leq H(\mathbf{XY}|\mathbf{WK}) && \text{z (11.9)} \\ &= H(\mathbf{X}|\mathbf{WK}) + H(\mathbf{Y}|\mathbf{WXK}) && \text{z (11.5)} \\ &\leq H(\mathbf{X}|\mathbf{WK}) + H(\mathbf{Y}|\mathbf{XK}) && \text{z (11.8)} \\ &= H(\mathbf{XW}) - H(\mathbf{K}) + (\mathbf{Y}|\mathbf{X}) - H(\mathbf{K}) && \text{z lematu 11.7} \\ &\leq H(\mathbf{X}) - H(\mathbf{K}) + (\mathbf{Y}|\mathbf{X}) - H(\mathbf{K}) && \text{z (11.7)} \\ &= H(\mathbf{XY}) - 2H(\mathbf{K}) && \text{z (11.4).} \end{aligned}$$

Stąd wynika teza twierdzenia.

|

**WNIOSZEK 11.10**

Niech  $\Gamma$  będzie strukturą dostępu spełniającą założenia twierdzenia 11.9. Wówczas, jeśli wszystkie klucze z  $\mathcal{K}$  są jednakowo prawdopodobne, to  $\rho \leq 2/3$ .

**DOWÓD** Jeśli klucze są jednakowo prawdopodobne, to

$$H(\mathbf{K}) = \log_2 |\mathcal{K}|.$$

Ponadto,

$$\begin{aligned} H(\mathbf{XY}) &\leq H(\mathbf{X}) + H(\mathbf{Y}) \\ &\leq \log_2 |\mathcal{S}(X)| + \log_2 |\mathcal{S}(Y)|. \end{aligned}$$

Z twierdzenia 11.9 wiemy, że

$$H(\mathbf{XY}) \geq 3H(\mathbf{K}),$$

więc

$$\log_2 |\mathcal{S}(X)| + \log_2 |\mathcal{S}(Y)| \geq 3 \log_2 |\mathcal{K}|.$$

Dalej, z definicji względnej miary informacji mamy

$$\rho \leq \frac{\log_2 |\mathcal{K}|}{\log_2 |\mathcal{S}(X)|}$$

oraz

$$\rho \leq \frac{\log_2 |\mathcal{K}|}{\log_2 |\mathcal{S}(Y)|}.$$

W konsekwencji

$$\begin{aligned} 3 \log_2 |\mathcal{K}| &\leq \log_2 |\mathcal{S}(X)| + \log_2 |\mathcal{S}(Y)| \\ &\leq \frac{\log_2 |\mathcal{K}|}{\rho} + \frac{\log_2 |\mathcal{K}|}{\rho} \\ &= 2 \frac{\log_2 |\mathcal{K}|}{\rho}. \end{aligned}$$

Tak więc  $\rho \leq 2/3$ . ■

Założenia twierdzenia 11.9 są spełnione dla struktur o numerach 5, 8, 12 i 13, zatem dla każdej z nich jest spełniona nierówność  $\rho^* \leq 2/3$ .

Udowodnimy także wynik dotyczący wartości  $\rho^*$  w przypadku struktur dostępu, których baza  $\Gamma_0$  jest grafem. Dowód wymaga wykazania, że każdy graf spójny nie będący grafem wielodzielnym zawiera podgraf indukowany na czterech wierzchołkach, izomorficzny z bazą struktury o numerze 5 lub 8. Jeśli  $G = (V, E)$  jest grafem ze zbiorem wierzchołków  $V$  i zbiorem krawędzi  $E$  oraz  $V_1 \subseteq V$ , to *podgrafem indukowanym*  $G[V_1]$  nazywamy graf  $(V_1, E_1)$ , w którym

$$E_1 = \{uv \in E : u, v \in V_1\}.$$

**TWIERDZENIE 11.11**

Niech  $G$  będzie grafem spójnym, który nie jest pełnym grafem wielodzielnym. Jeśli  $\Gamma(G)$  jest strukturą dostępu, będącą domknięciem  $E$  (gdzie  $E$  jest zbiorem krawędzi grafu  $G$ ), to  $\rho^*(\Gamma(G)) \leq 2/3$ .

**DOWÓD** Wykażemy najpierw, że każdy graf spójny, który nie jest pełnym grafem wielodzielnym, zawiera cztery wierzchołki  $w, x, y, z$ , takie że graf indukowany  $G[w, x, y, z]$  jest izomorficzny albo z bazą struktury numer 5, albo z bazą struktury numer 8. Niech  $G^C$  oznacza dopełnienie grafu  $G$ . Jeśli  $G$  nie jest pełnym grafem wielodzielnym, to istnieją trzy wierzchołki  $x, y, z$ , takie że  $xy, yz \in E(G^C)$  i  $xz \in E(G)$ . Niech

$$d = \min\{d_G(y, x), d_G(y, z)\},$$

gdzie  $d_G$  oznacza długość najkrótszej ścieżki (w  $G$ ) między dwoma wierzchołkami. Z definicji wynika więc, że  $d \geq 2$ . Bez utraty ogólności możemy przyjąć, że  $d = d_G(y, x)$  (ze względu na symetrię). Niech

$$y_0, y_1, \dots, y_{d-1}, x$$

będzie ścieżką w  $G$ , w której  $y_0 = y$ . Mamy wówczas

$$y_{d-2}z, y_{d-2}x \in E(G^C)$$

oraz

$$y_{d-2}y_{d-1}, y_{d-1}x, x, z \in E(G).$$

Wynika stąd, że graf  $G[y_{d-2}, y_{d-1}, x, z]$  jest izomorficzny z bazą struktury dostępu o numerze 5 lub 8, co należało udowodnić.

Możemy zatem przyjąć, że znaleźliśmy cztery wierzchołki  $w, x, y, z$ , takie że graf indukowany  $G[w, x, y, z]$  jest izomorficzny z bazą struktury o numerze 5 lub 8. Niech teraz  $\mathcal{F}$  będzie dowolnym schematem realizującym strukturę dostępu  $\Gamma(G)$ . Jeśli ograniczymy dziedzinę reguł dystrybucji do zbioru  $\{w, x, y, z\}$ , to otrzymamy schemat  $\mathcal{F}'$  realizujący strukturę 5 lub 8. Rzecz jasna,  $\rho(\mathcal{F}') \geq \rho(\mathcal{F})$ , a ponieważ  $\rho(\mathcal{F}') \leq 2/3$ , to i  $\rho(\mathcal{F}) \leq 2/3$ . To kończy dowód. ■

Dla pełnych grafów wielodzielnych mamy  $\rho^* = 1$ , zatem twierdzenie 11.11 mówi, że dla żadnej struktury dostępu będącej domknięciem zbioru krawędzi grafu spójnego nie jest spełniona podwójna nierówność  $2/3 < \rho^* < 1$ .

## 11.8. Konstrukcja przez rozkład

Pozostały nam jeszcze do rozważenia cztery struktury z tablicy 11.1. Moglibyśmy oczywiście uzyskać dla nich schematy za pomocą konstrukcji opartej na obwodzie monotonicznym, ale w najlepszym razie otrzymalibyśmy dla każdej

z nich względną miarę informacji  $\rho = 1/2$ . Dla struktur o numerach 5 i 12 możemy uzyskać  $\rho = 1/2$  na podstawie obwodu odpowiadającego dysjunktywnej postaci normalnej; takie obwody dają dla struktur o numerach 8 i 13 wartość  $\rho = 1/3$ , istnieją jednak inne obwody monotoniczne, które umożliwiają osiągnięcie  $\rho = 1/2$ . W rzeczywistości można dla każdej z tych czterech struktur zbudować schemat z wartością  $\rho = 2/3$ . W konstrukcji prowadzącej do takiego wyniku podstawowymi blokami, z których tworzy się większe schematy, są schematy idealne.

Przedstawimy tu konstrukcję tego typu, którą nazwiemy „konstrukcją przez rozkład”. Potrzebne będzie do tego następujące ważne pojęcie.

### DEFINICJA 11.6

Niech  $\Gamma$  będzie strukturą dostępu z bazą  $\Gamma_0$ , a  $\mathcal{K}$  danym zbiorem kluczy. *Idealnym  $\mathcal{K}$ -rozkładem*  $\Gamma_0$  jest zbiór  $\{\Gamma_1, \dots, \Gamma_n\}$  spełniający następujące warunki:

- (1)  $\Gamma_k \subseteq \Gamma_0$  dla  $1 \leq k \leq n$ ,
- (2)  $\bigcup_{k=1}^n \Gamma_k = \Gamma_0$ ,
- (3) dla struktury dostępu z bazą  $\Gamma_k$  (dla każdego  $1 \leq k \leq n$ ) istnieje idealny schemat ze zbiorem kluczy  $\mathcal{K}$  na podzbiorze uczestników

$$\mathcal{P}_k = \bigcup_{B \in \Gamma_k} B.$$

Dla danego idealnego  $\mathcal{K}$ -rozkładu struktury dostępu  $\Gamma$  możemy bez trudu zbudować doskonały tajny schemat współużytkowania, jak to opisuje następujące twierdzenie.

### TWIERDZENIE 11.12

Niech  $\Gamma$  będzie strukturą dostępu z bazą  $\Gamma_0$ , a  $\mathcal{K}$  danym zbiorem kluczy. Założymy, że  $\{\Gamma_1, \dots, \Gamma_n\}$  jest idealnym  $\mathcal{K}$ -rozkładem  $\Gamma$ . Dla każdego uczestnika  $P_i$  definiujemy

$$R_i = |\{k : P_i \in \mathcal{P}_k\}|.$$

Wówczas istnieje doskonały tajny schemat współużytkowania realizujący strukturę  $\Gamma$ , dla którego  $\rho = 1/R$ , gdzie

$$R = \max\{R_i : 1 \leq i \leq w\}.$$

**DOWÓD** Dla  $1 \leq k \leq n$  mamy idealny schemat realizujący strukturę dostępu z bazą  $\Gamma_k$ , zbiorem kluczy  $\mathcal{K}$  i zbiorem reguł dystrybucji  $\mathcal{F}^k$ . Zbudujmy teraz schemat realizujący  $\Gamma$  ze zbiorem kluczy  $\mathcal{K}$ . Zbiór reguł dystrybucji  $\mathcal{F}$  powstaje tak, że gdy  $D$  chce rozdzielić klucz  $K$ , dla każdego  $1 \leq k \leq n$  wybiera losowo regułę  $f^k \in \mathcal{F}_K^k$  i przypisuje wynikające z niej udziały uczestnikom ze zbioru  $\mathcal{P}_k$ .

Nie będziemy dowodzić, że schemat jest doskonały, łatwo natomiast można obliczyć względną miarę informacji otrzymanego systemu. Każdy ze schematów składowych jest idealny, zatem

$$|\mathcal{S}(P_i)| = |\mathcal{K}|^{R_i},$$

dla  $1 \leq i \leq w$ . W rezultacie

$$\rho_i = \frac{1}{R_i}$$

oraz

$$\rho = \frac{1}{\max\{R_i : 1 \leq i \leq w\}},$$

co należało dowieść. ■

Mimo użyteczności twierdzenia 11.12 często bardziej przydatne okazuje się jego uogólnienie, w którym mamy  $\ell$  idealnych  $\mathcal{K}$ -rozkładów zamiast jednego. Każdy z nich jest używany do rozdziału klucza wybranego ze zbioru  $\mathcal{K}$ . Budujemy zatem schemat ze zbiorem kluczy  $\mathcal{K}^\ell$ ; konstrukcję takiego schematu i jego względową miarę informacji opisuje następujące twierdzenie.

### **TWIERDZENIE 11.13** (konstrukcja przez rozkład)

Niech  $\Gamma$  będzie strukturą dostępu z bazą  $\Gamma_0$ , a  $\mathcal{K}$  danym zbiorem kluczy. Dalej, niech  $\ell \geq 1$  będzie liczbą całkowitą; założymy, że dla  $1 \leq j \leq \ell$  zbiór  $\mathcal{D}_j = \{\Gamma_{j,1}, \dots, \Gamma_{j,n_j}\}$  jest idealnym rozkładem  $\Gamma_0$ . Niech  $\mathcal{P}_{j,k}$  oznacza zbiór uczestników struktury dostępu  $\Gamma_{j,k}$ . Dla każdego uczestnika  $P_i$  definiujemy

$$R_i = \sum_{j=1}^{\ell} |\{k : P_i \in \mathcal{P}_{j,k}\}|.$$

Wówczas istnieje doskonały tajny schemat współużytkowania realizujący  $\Gamma$  ze względową miarą informacji  $\rho = \ell/R$ , gdzie

$$R = \max\{R_i : 1 \leq i \leq w\}.$$

**DOWÓD** Dla  $1 \leq j \leq \ell$  i  $1 \leq k \leq n$  mamy idealny schemat realizujący strukturę dostępu z bazą  $\Gamma_{j,k}$ , dla którego zbiorem kluczy jest  $\mathcal{K}$ , a zbiorem reguł dystrybucji  $\mathcal{F}^{j,k}$ . Zbudujemy schemat ze zbiorem kluczy  $\mathcal{K}^\ell$  realizujący  $\Gamma$ . Zbiór reguł dystrybucji  $\mathcal{F}$  powstaje tak, że gdy  $D$  chce rozdzielić klucz  $K = (K_1, \dots, K_\ell)$ , dla każdego  $1 \leq j \leq \ell$  i  $1 \leq k \leq n$  wybiera losowo regułę  $f^{j,k} \in \mathcal{F}_{K_j}^{j,k}$  i przypisuje wynikające z niej udziały uczestnikom ze zbioru  $\mathcal{P}_{j,k}$ .

Względną miarę informacji można obliczyć podobnie jak w twierdzeniu 11.12. ■

Popatrzmy na kilka przykładów.

**Przykład 11.10**

Zajmiemy się strukturą dostępu numer 5. Jej bazą jest graf, który nie jest pełnym grafem wielodzielnym, więc na mocy twierdzenia 11.11 zachodzi nierówność  $\rho^* \leq 2/3$ .

Niech  $p$  będzie liczbą pierwszą. Rozpatrzmy dwa idealne  $\mathbb{Z}_p$ -rozkłady:

$$\mathcal{D}_1 = \{\Gamma_{1,1}, \Gamma_{1,2}\},$$

gdzie

$$\Gamma_{1,1} = \{\{P_1, P_2\}\}$$

$$\Gamma_{1,2} = \{\{P_2, P_3\}, \{P_3, P_4\}\}$$

oraz

$$\mathcal{D}_2 = \{\Gamma_{2,1}, \Gamma_{2,2}\},$$

gdzie

$$\Gamma_{2,1} = \{\{P_1, P_2\}, \{P_2, P_3\}\}$$

$$\Gamma_{2,2} = \{\{P_3, P_4\}\}.$$

Każdy z tych rozkładów składa się z grafów  $K_2$  i  $K_{1,2}$ , zatem są to rzeczywiście idealne  $\mathbb{Z}_p$ -rozkłady. W obu przypadkach otrzymujemy schemat ze względną miarą informacji  $\rho = 1/2$ . Jeśli jednak „połączymy” je zgodnie z twierdzeniem 11.13 (dla  $\ell = 2$ ), otrzymamy schemat, dla którego  $\rho = 2/3$ , co jest wartością optymalną.

Opiszemy teraz jedną z możliwych implementacji tego schematu, odwołującą się do twierdzenia 11.5. Rozdający  $D$  wybiera niezależnie cztery losowe elementy z  $\mathbb{Z}_p$ , powiedzmy:  $b_{11}$ ,  $b_{12}$ ,  $b_{21}$  oraz  $b_{22}$ . Jeśli kluczem jest  $(K_1, K_2) \in (\mathbb{Z}_p)^2$ , to  $D$  rozdziela udziały w następujący sposób:

- (1)  $P_1$  dostaje  $b_{11}, b_{21}$ .
- (2)  $P_2$  dostaje  $b_{11} + K_1, b_{12}, b_{21} + K_2$ .
- (3)  $P_3$  dostaje  $b_{12} + K_1, b_{21}, b_{22}$ .
- (4)  $P_4$  dostaje  $b_{12}, b_{22} + K_2$ .

(Wszystkie obliczenia odbywają się w  $\mathbb{Z}_p$ ). □

**Przykład 11.11**

Rozpatrzmy teraz strukturę numer 8. Znowu na mocy twierdzenia 11.11 mamy  $\rho^* \leq 2/3$ , a połączenie dwóch odpowiednich idealnych rozkładów daje w wyniku (optymalny) schemat, dla którego  $\rho = 2/3$ .

Niech  $\mathcal{K} = \mathbb{Z}_p$  dla dowolnej liczby pierwszej  $p \geq 3$ . Określmy następujące dwa idealne  $\mathcal{K}$ -rozkłady:

$$\mathcal{D}_1 = \{\Gamma_{1,1}, \Gamma_{1,2}\},$$

gdzie

$$\Gamma_{1,1} = \{\{P_1, P_2\}\}$$

$$\Gamma_{1,2} = \{\{P_2, P_3\}, \{P_2, P_4\}, \{P_3, P_4\}\}$$

oraz

$$\mathcal{D}_2 = \{\Gamma_{2,1}, \Gamma_{2,2}\},$$

gdzie

$$\Gamma_{2,1} = \{\{P_1, P_2\}, \{P_2, P_3\}, \{P_2, P_4\}\}$$

$$\Gamma_{2,2} = \{\{P_3, P_4\}\}.$$

Rozkład  $\mathcal{D}_1$  składa się z grafów  $K_2$  i  $K_3$ , rozkład  $D_2$  – z grafów  $K_2$  i  $K_{1,3}$ , zatem oba są idealnymi  $\mathcal{K}$ -rozkładami. Stosując twierdzenie 11.13 dla  $\ell = 2$ , otrzymujemy schemat, dla którego  $\rho = 2/3$ .

Opiszymy implementację tego schematu, odwołującą się do twierdzenia 11.5. Rozdający  $D$  wybiera niezależnie cztery losowe elementy z  $\mathbb{Z}_p$ , powiedzmy:  $b_{11}$ ,  $b_{12}$ ,  $b_{21}$  oraz  $b_{22}$ . Jeśli kluczem jest  $(K_1, K_2) \in (\mathbb{Z}_p)^2$ , to  $D$  rozdziela udziały w następujący sposób:

- (1)  $P_1$  dostaje  $b_{11} + K_1, b_{21} + K_2$ .
- (2)  $P_2$  dostaje  $b_{11}, b_{12}, b_{21}$ .
- (3)  $P_3$  dostaje  $b_{12} + K_1, b_{21} + K_2, b_{22}$ .
- (4)  $P_4$  dostaje  $b_{12} + 2K_1, b_{21} + K_2, b_{22} + K_2$ .

(Wszystkie obliczenia odbywają się w  $\mathbb{Z}_p$ ). □

Wyjaśniliśmy dotąd wszystkie dane zapisane w tablicy 11.1, z wyjątkiem wartości  $\rho^*$  dla struktur o numerach 12 i 13. Wartości te wynikają z ogólniejszej wersji konstrukcji przez rozkład, którą tu nie będziemy się zajmować; patrz uwagi niżej.

## 11.9. Uwagi i bibliografia

Schematy progowe wymyślili niezależnie od siebie Blakley [BL79] i Shamir [SH79]. Po raz pierwszy badaniem schematów współużytkowania dla ogólnych struktur dostępu zajęli się Ito, Saito i Nishizeki [ISN87]; w podrozdziale 11.2 zaprezentowaliśmy podejście zaproponowane przez Benaloha i Leichtera [BL90]. Konstrukcja oparta na przestrzeni liniowej pochodzi od Brickella [BR89A]. Ograniczenie związane z entropią omawiane w podrozdziale 11.7 zostało udowodnione

przez Capocelliego i innych [CDGV93]; niektóre inne informacje zawarte w tym podrozdziale można znaleźć w pracy Blunda i innych [BDSV93].

Podkreślimy w tym rozdziale liniowo-algebraiczne i kombinatoryczne podejście do tajnego współużytkowania. Interesujące powiązania z teorią matroidów można znaleźć w pracy Brickella i Davenporta [BD91]. Tajne schematy współużytkowania można budować także za pomocą metod geometrycznych; znaczące badania w tym zakresie przeprowadził Simmons. Warto wskazać [Si92A] jako przegląd metod geometrycznych w dziedzinie tajnego współużytkowania. Dalsze omówienie tych tematów, podobnie jak konstrukcje schematów ze względną miarą informacji  $2/3$  dla struktur dostępu o numerach 12 i 13, można znaleźć w monograficznym artykule Stinsona [ST92A].

## Ćwiczenia

11.1. Napisz program komputerowy obliczający klucz dla schematu progowego Shamira o parametrach  $t, w$  nad  $\mathbb{Z}_p$ . Dokładniej, dla danych  $t$  publicznych wartości współrzędnych  $x_1, x_2, \dots, x_t$  i  $t$  wartości współrzędnych  $y_1, \dots, y_t$  oblicz wartość klucza. Użyj interpolacji Lagrange'a, ponieważ łatwiej ją zaprogramować.

- (a) Przetestuj swój program z parametrami  $p = 31847$ ,  $t = 5$  i  $w = 10$  na następujących udziałach:

$$\begin{array}{ll} x_1 = 413 & y_1 = 25439 \\ x_2 = 432 & y_2 = 14847 \\ x_3 = 451 & y_3 = 24780 \\ x_4 = 470 & y_4 = 5910 \\ x_5 = 489 & y_5 = 12734 \\ x_6 = 508 & y_6 = 12492 \\ x_7 = 527 & y_7 = 12555 \\ x_8 = 546 & y_8 = 28578 \\ x_9 = 565 & y_9 = 20806 \\ x_{10} = 584 & y_{10} = 21462 \end{array}$$

Sprawdź, że różne pięcioelementowe podzbiory prowadzą do tego samego klucza.

- (b) Po ustaleniu klucza oblicz udział, który należałoby przypisać uczestnikowi, który dostał pierwszą współrzędną równą 10000. Zauważ, że można to zrobić bez obliczenia całego tajnego wielomianu  $a(x)$ .

11.2. Nieużciwy rozdający mógłby rozdać w schemacie progowym Shamira „mylące” udziały, a mianowicie takie, które dla różnych  $t$ -elementowych podzbiorów dawałyby różne klucze. Znając udziały wszystkich  $w$  uczestników, moglibyśmy sprawdzić ich niesprzeczność przez obliczenie klucza dla każdego z  $\binom{w}{t}$   $t$ -elementowych podzbiorów zbioru uczestników i porównanie otrzymanych wyników. Spróbuj opisać bardziej efektywny sposób sprawdzenia niesprzeczności udziałów.

11.3. Użyj konstrukcji opartej na obwodzie monotonicznym, by dla każdej ze struktur dostępu ze wskazaną bazą  $\Gamma_0$  zbudować tajny schemat współużytkowania ze względną miarą informacji  $\rho = 1/3$ .

- (a)  $\Gamma_0 = \{\{P_1, P_2\}, \{P_2, P_3\}, \{P_2, P_4\}, \{P_3, P_4\}\}.$
- (b)  $\Gamma_0 = \{\{P_1, P_3, P_4\}, \{P_1, P_2\}, \{P_2, P_3\}, \{P_2, P_4\}\}.$
- (c)  $\Gamma_0 = \{\{P_1, P_2\}, \{P_1, P_3\}, \{P_2, P_3, P_4\}, \{P_2, P_4, P_5\}, \{P_3, P_4, P_5\}\}.$
- 11.4. Użyj konstrukcji opartej na przestrzeni liniowej, by dla każdej ze struktur dostępu ze wskazaną bazą  $\Gamma_0$  zbudować schemat idealny.
- (a)  $\Gamma_0 = \{\{P_1, P_2, P_3\}, \{P_1, P_2, P_4\}, \{P_3, P_4\}\}.$
- (b)  $\Gamma_0 = \{\{P_1, P_2, P_3\}, \{P_1, P_2, P_4\}, \{P_1, P_3, P_4\}\}.$
- (c)  $\Gamma_0 = \{\{P_1, P_2\}, \{P_1, P_3\}, \{P_2, P_3\}, \{P_1, P_4, P_5\}, \{P_2, P_4, P_5\}\}.$
- 11.5. Użyj konstrukcji przez rozkład, by zbudować schemat dla każdej ze struktur dostępu ze wskazaną bazą  $\Gamma_0$  i względną miarą informacji  $\rho$ .
- (a)  $\Gamma_0 = \{\{P_1, P_3, P_4\}, \{P_1, P_2\}, \{P_2, P_3\}\}, \rho = 3/5.$
- (b)  $\Gamma_0 = \{\{P_1, P_3, P_4\}, \{P_1, P_2\}, \{P_2, P_3\}, \{P_2, P_4\}\}, \rho = 4/7.$

# 12

---

## Generowanie liczb pseudolosowych

---

### 12.1. Wprowadzenie i przykłady

Kryptografia obfituje w sytuacje, w których konieczne jest utworzenie losowych liczb, ciągów bitów itp. Na przykład, klucze kryptograficzne wybiera się losowo z danej przestrzeni kluczów, podczas realizacji wielu protokołów trzeba wygenerować liczby losowe. Losowe wybieranie liczb za pomocą rzutu monetą lub innego procesu fizycznego jest czasochłonne i kosztowne, dlatego w praktyce często używa się *generatora bitów pseudolosowych* (czyli PRBG od ang. *pseudo-random bit generator*). Do generatora bitów pseudolosowych wprowadza się na początek krótki losowy ciąg bitów („zalążek”), który zostaje w nim rozszerzony do znacznie dłuższego, „losowo wyglądającego” ciągu bitów. W ten sposób PRBG zmniejsza liczbę losowych bitów niezbędnych w danym zastosowaniu.

Formalnie możemy to ująć w postaci następującej definicji.

#### DEFINICJA 12.1

Niech  $k, \ell$  będą dodatnimi liczbami całkowitymi, takimi że  $\ell \geq k+1$  (gdzie  $\ell$  jest wyznaczona wielomianową zależnością od  $k$ ). *Generatorem bitów pseudolosowych o parametrach  $k, \ell$*  (w skrócie  $(k, \ell)$ -PRBG) nazywamy funkcję  $f : (\mathbb{Z}_2)^k \rightarrow (\mathbb{Z}_2)^\ell$  obliczalną w czasie wielomianowo zależnym od  $k$ . Argument  $s_0 \in (\mathbb{Z}_2)^k$  nazywamy *zalążkiem*, wartość  $f(s_0) \in (\mathbb{Z}_2)^\ell$  zaś *pseudolosowym ciągiem bitów*.

Funkcja  $f$  jest deterministyczna, więc ciąg bitów  $f(s_0)$  zależy jedynie od zalążka. Chcemy, by przy losowym wyborze zalążka pseudolosowy ciąg bitów  $f(s_0)$  „wyglądał” jak ciąg rzeczywiście losowy. Trudno tu sformułować precyzyjną definicję, spróbujemy jednak w dalszej części rozdziału podać intuicyjny opis tego pojęcia.

Zobaczmy przykład, będący argumentem za badaniem tego typu generatorów bitów pseudolosowych. W rozdziale 2 rozpatrywaliśmy pojęcie tajności doskonałej. Jedną z реализациj takiej tajności okazał się szyfr z kluczem jednorazowym, w którym zarówno tekst jawnny, jak i klucz są ciągami bitów określonej długości,

a tekst zaszyfrowany powstaje w wyniku obliczenia różnicy symetrycznej tekstu jawnego i klucza. Praktyczna niedogodność szyfru z kluczem jednorazowym polega na tym, że w celu zapewnienia tajności doskonałej klucz, który ma być wygenerowany losowo i przesłany za pośrednictwem bezpiecznego kanału, musi mieć tę samą długość, co tekst jawnny. Generator bitów pseudolosowych tą niedogodność łagodzi. Przypuśćmy, że Alicja i Bolek użgadniają PRBG i przesyłają załączek bezpiecznym kanałem. Wtedy oboje mogą obliczyć ten sam ciąg pseudolosowych bitów, który może teraz posłużyć za szyfr z kluczem jednorazowym. W tej sytuacji załączek pełni funkcję klucza, a PRBG występuje jako generator klucza strumieniowego w szyfrze strumieniowym.

Dla ilustracji pojęć, które będziemy dalej badali, przedstawimy kilka dobrze znanych generatorów bitów pseudolosowych. Zauważmy najpierw, że liniowy rejestr przesuwający ze sprzężeniem zwrotnym (LFSR), opisany w punkcie 1.1.7, działa tak jak PRBG. Z jednego  $k$ -bitowego załącznika można za pomocą  $k$ -bitowego LFSR utworzyć aż  $2^k - k - 1$  nowych bitów, zanim wystąpi powtórzenie. PRBG uzyskany za pomocą LFSR jest bardzo niepewny: stwierdziliśmy już w punkcie 1.2.5, że wiedza o dowolnych  $2k$  kolejnych bitach pozwala ustalić wartość załącznika, a stąd również całego ciągu. (Co prawda nie zdefiniowaliśmy dotąd bezpieczeństwa PRBG, ale, rzecz jasna, istnienie możliwości tego typu ataku oznacza, że generator bezpieczny nie jest!).

Inny znany (choć również niepewny) generator bitów pseudolosowych, zwany **liniowym generatorem kongruencyjnym**, przedstawiamy na rysunku 12.1. Oto prosty przykład dla ilustracji.

Niech  $M \geq 2$  będzie liczbą całkowitą i  $1 \leq a, b \leq M - 1$ . Niech  $k = \lceil \log_2 M \rceil$  oraz  $k + 1 \leq \ell \leq M - 1$ .

Dla załącznika  $s_0$ , gdzie  $0 \leq s_0 \leq M - 1$ , określamy

$$s_i = (as_{i-1} + b) \bmod M$$

dla  $1 \leq i \leq \ell$ , a następnie definiujemy

$$f(s_0) = (z_1, z_2, \dots, z_\ell),$$

gdzie

$$z_i = s_i \bmod 2,$$

$1 \leq i \leq \ell$ . Wówczas  $f$  jest  $(k, \ell)$ -liniowym generatorem kongruencyjnym.

RYSUNEK 12.1. Liniowy generator kongruencyjny

**Przykład 12.1**

Jeśli w liniowym generatorze kongruencyjnym przyjmiemy  $M = 31$ ,  $a = 3$  oraz  $b = 5$ , otrzymamy (5,10)-PRBG. Rozważmy przekształcenie  $s \mapsto 3s + 5 \pmod{31}$ ; wówczas  $13 \mapsto 13$ , a pozostałe 30 reszt jest permutowanych w cyklu o długości 30, mianowicie: 0, 5, 20, 3, 14, 16, 22, 9, 1, 8, 29, 30, 2, 11, 7, 26, 21, 6, 23, 12, 10, 4, 17, 25, 18, 28, 27, 24, 15, 19. Jeśli założkiem jest liczba różna od 13, to wskazuje on początkowy element w tym cyklu, podczas gdy kolejne 10 elementów, zredukowanych modulo 2 stanowi wynikowy ciąg pseudolosowy.

W tablicy 12.1 umieszczone zostały 31 możliwe pseudolosowe ciągi bitów wyprodukowane przez taki generator. □

**TABLICA 12.1. Ciągi bitów uzyskane za pomocą liniowego generatora kongruencyjnego**

| Załązek | Ciąg       |
|---------|------------|
| 0       | 1010001101 |
| 1       | 0100110101 |
| 2       | 1101010001 |
| 3       | 0001101001 |
| 4       | 1100101101 |
| 5       | 0100011010 |
| 6       | 1000110010 |
| 7       | 0101000110 |
| 8       | 1001101010 |
| 9       | 1010011010 |
| 10      | 0110010110 |
| 11      | 1010100011 |
| 12      | 0011001011 |
| 13      | 1111111111 |
| 14      | 0011010011 |
| 15      | 1010100011 |
| 16      | 0110100110 |
| 17      | 1001011010 |
| 18      | 0101101010 |
| 19      | 0101000110 |
| 20      | 1000110100 |
| 21      | 0100011001 |
| 22      | 1101001101 |
| 23      | 0001100101 |
| 24      | 1101010001 |
| 25      | 0010110101 |
| 26      | 1010001100 |
| 27      | 0110101000 |
| 28      | 1011010100 |
| 29      | 0011010100 |
| 30      | 0110101000 |

Do konstrukcji generatora bitów pseudolosowych możemy wykorzystać kilka pojęć zdefiniowanych w poprzednich rozdziałach. Na przykład, tryb wyjściowy sprzężenia zwrotnego systemu DES, opisany w punkcie 3.4.1, możemy traktować jako generator bitów pseudolosowych; co więcej, wydaje się, że będzie to generator obliczeniowo bezpieczny.

Inny sposób tworzenia bardzo szybkich generatorów bitów pseudolosowych polega na łączeniu liniowych rejestrów przesuwających ze sprzężeniem zwrotnym, tak by otrzymywane wyniki miały mniej liniowy charakter. Jedna z takich metod, zaproponowana przez Coppersmitha, Krawczyka i Mansoura, nosi nazwę **generatora obcinającego** (ang. *shrinking generator*). Założymy, że mamy dwa liniowe rejesty przesuwające ze sprzężeniem zwrotnym, jeden stopnia  $k_1$ , drugi stopnia  $k_2$ . Do uruchomienia obu będziemy potrzebować załączka o  $k_1 + k_2$  bitach. Pierwszy z nich wyprodukuje ciąg bitów  $a_1, a_2, \dots$ , drugi – ciąg  $b_1, b_2, \dots$ . Określamy wówczas ciąg pseudolosowych bitów  $z_1, z_2, \dots$  zgodnie z regułą

$$z_i = a_{i_k},$$

gdzie  $i_k$  oznacza pozycję  $k$ -tej jedynki w ciągu  $b_1, b_2, \dots$ . Otrzymany ciąg bitów składa się więc z podcięgu ciągu bitów wyprodukowanych przez pierwszy rejestr. Taka metoda generowania pseudolosowych bitów jest bardzo szybka i odporna na wiele znanych rodzajów ataku, nie widać jednak żadnego sposobu udowodnienia, że jest to metoda bezpieczna.

W pozostałej części rozdziału zajmiemy się generatorami bitów pseudolosowych, dla których można przy pewnych rozsądnych założeniach obliczeniowych udowodnić, że zapewniają należyté bezpieczeństwo. Istnieją PRBG oparte na

Niech  $p$  i  $q$  będą dwiema  $k/2$ -bitowymi liczbami pierwszymi i  $n = pq$ . Przyjmijmy, że  $b$  jest liczbą wybraną tak, aby  $NWD(b, \phi(n)) = 1$ . Jak zwykle,  $n$  i  $b$  są publiczne, podczas gdy  $p$  i  $q$  są tajne.

Załączek  $s_0$  jest dowolnym elementem  $\mathbb{Z}_n^*$ , zatem  $s_0$  ma  $k$  bitów. Dla  $i \geq 1$  zdefiniujmy

$$s_{i+1} = s_i^b \bmod n,$$

a następnie

$$f(s_0) = (z_1, z_2, \dots, z_\ell),$$

gdzie

$$z_i = s_i \bmod 2$$

dla  $1 \leq i \leq \ell$ . Wówczas  $f$  jest  $(k, \ell)$ -generatorem RSA.

RYSUNEK 12.2. Generator RSA

fundamentalnych problemach faktoryzacji (związkowych z systemem kryptograficznym RSA z publicznym kluczem) oraz na problemie logarytmu dyskretnego. Na rysunku 12.2 jest pokazany PRBG używający funkcji szyfrującej RSA, natomiast PRBG związany z problemem logarytmu dyskretnego omawiamy w ćwiczeniach.

Podamy teraz przykład **generatora RSA**.

### Przykład 12.2

Niech  $n = 91261 = 263 \cdot 347$ ,  $b = 1547$  oraz  $s_0 = 75364$ . Pierwszych 20 bitów wyprodukowanych przez generator RSA znajduje się w tablicy 12.2. Ciągiem bitów otrzymanym z tego załącznika jest więc ciąg

10000111011110011000.

□

**TABLICA 12.2. Bitы wytworzzone za pomocą generatora RSA**

| $i$ | $s_i$ | $z_i$ |
|-----|-------|-------|
| 0   | 75634 |       |
| 1   | 31483 | 1     |
| 2   | 31238 | 0     |
| 3   | 51968 | 0     |
| 4   | 39796 | 0     |
| 5   | 28716 | 0     |
| 6   | 14089 | 1     |
| 7   | 5923  | 1     |
| 8   | 44891 | 1     |
| 9   | 62284 | 0     |
| 10  | 11889 | 1     |
| 11  | 43467 | 1     |
| 12  | 71215 | 1     |
| 13  | 10401 | 1     |
| 14  | 77444 | 0     |
| 15  | 56794 | 0     |
| 16  | 78147 | 1     |
| 17  | 72137 | 1     |
| 18  | 89592 | 0     |
| 19  | 29022 | 0     |
| 20  | 13356 | 0     |

## 12.2. Nierozróżnialne rozkłady prawdopodobieństwa

Od generatora liczb pseudolosowych oczekuje się dwóch zasadniczych cech: powinien być szybki (a więc obliczalny w czasie wielomianowym względem  $k$ ) oraz bezpieczny. Oczywiście często te dwa wymogi pozostają w konflikcie. Generatory

oparte na liniowych kongruencjach lub liniowych rejestrach przesuwających ze sprzężeniem zwotnym są rzeczywiście bardzo szybkie i bardzo przydatne przy symulacjach, natomiast mało bezpieczne w zastosowaniach kryptograficznych.

Spróbujmy teraz nieco precyzyjniej określić pojęcie „bezpieczeństwa” generatora bitów pseudolosowych. Intuicyjnie, ciąg  $k^m$  bitów uzyskanych za pomocą generatora bitów pseudolosowych powinien wydawać się „losowy”. Inaczej mówiąc, odróżnienie ciągu  $\ell$  pseudolosowych bitów wygenerowanych przez PRBG od rzeczywiście losowego ciągu  $\ell$  bitów nie powinno być możliwe w czasie wielomianowym względem  $k$  (lub równoważnie, względem  $\ell$ ).

Takie rozważania prowadzą do pojęcia rozróżnialności rozkładów prawdopodobieństwa. Oto jego definicja.

### DEFINICJA 12.2

Niech  $p_0$  i  $p_1$  będą dwoma rozkładami prawdopodobieństwa na zbiorze  $(\mathbb{Z}_2)^\ell$  wszystkich  $\ell$ -wyrazowych ciągów bitów. Ponadto, niech  $\mathbf{A} : (\mathbb{Z}_2)^\ell \rightarrow \{0, 1\}$  będzie algorytmem probabilistycznym działającym w czasie wielomianowym (jako funkcja zmiennej  $\ell$ ). Dla  $j = 0, 1$  określamy

$$E_{\mathbf{A}}(p_j) = \sum_{(z_1, \dots, z_\ell) \in (\mathbb{Z}_2)^\ell} p_j(z_1, \dots, z_\ell) \cdot p(\mathbf{A}(z_1, \dots, z_\ell) = 1 | (z_1, \dots, z_\ell)).$$

Niech  $\epsilon > 0$ . Mówimy, że  $\mathbf{A}$  rozróżnia  $p_0$  i  $p_1$  na poziomie  $\epsilon$ , jeśli

$$|E_{\mathbf{A}}(p_0) - E_{\mathbf{A}}(p_1)| \geq \epsilon.$$

Jeśli istnieje algorytm rozróżniający  $p_0$  i  $p_1$  na poziomie  $\epsilon$ , to rozkłady  $p_0$  i  $p_1$  nazywamy  $\epsilon$ -rozróżnialnymi.

**UWAGA** Jeśli  $\mathbf{A}$  jest algorytmem deterministycznym, to prawdopodobieństwa warunkowe

$$p(\mathbf{A}(z_1, \dots, z_\ell) = 1 | (z_1, \dots, z_\ell))$$

mają zawsze wartość 0 lub 1. ■

Spróbujmy przedstawić co intuicyjnie oznacza powyższa definicja. Algorytm  $\mathbf{A}$  stara się ustalić, czy bardziej prawdopodobne jest to, że dany ciąg bitów  $(z_1, \dots, z_\ell)$  wynikł z rozkładu prawdopodobieństwa  $p_1$ , czy raczej to, że powstał na podstawie rozkładu  $p_0$ . Mogą w nim wystąpić liczby losowe, co oznacza, że może on być probabilistyczny. Wynik wykonania  $\mathbf{A}(z_1, \dots, z_\ell)$  określa ten z dwóch rozkładów, który został przez algorytm wskazany jako bardziej prawdopodobne źródło ciągu  $(z_1, \dots, z_\ell)$ . Liczba  $E_{\mathbf{A}}(p_j)$  reprezentuje średnią (czyli oczekiwana) wartość wyniku wykonania algorytmu  $\mathbf{A}$  dla rozkładu prawdopodobieństwa  $p_j$ , dla  $j = 0, 1$ . Wyznacza ją suma po wszystkich możliwych ciągach  $(z_1, \dots, z_\ell)$ , w której każdy ze składników jest iloczynem prawdopodobieństwa

ciągu  $(z_1, \dots, z_\ell)$  i prawdopodobieństwa tego, że  $\mathbf{A}$  odpowie „1”, gdy otrzyma ten ciąg na wejściu. Algorytm  $\mathbf{A}$  rozróżnia dane dwa rozkłady na poziomie  $\epsilon$ , jeśli obie wartości oczekiwane różnią się o co najmniej  $\epsilon$ .

Jaki to ma związek z generatorem bitów pseudolosowych? Rozważmy  $\ell$ -wyrazowe ciągi bitów. Jest ich  $2^\ell$ . Jeśli bity są wybierane niezależnie i losowo, to każdy z tych  $2^\ell$  ciągów występuje z tym samym prawdopodobieństwem  $1/2^\ell$ . Tak więc rzeczywiście losowy ciąg odpowiada rozkładowi nad zbiorem wszystkich  $\ell$ -wyrazowych ciągów bitów, w którym każdemu ciągowi jest przypisane to samo prawdopodobieństwo. Niech  $p_0$  oznacza taki rozkład prawdopodobieństwa.

Zajmijmy się teraz ciągami generowanymi przez PRBG. Założymy, że wybieramy losowo  $k$ -bitowy załączek, a następnie stosujemy nasz generator do uzyskania ciągu  $\ell$  bitów. Powstaje wówczas rozkład prawdopodobieństwa na zbiorze wszystkich ciągów  $\ell$ -bitowych; oznaczmy go przez  $p_1$ . (Dla większej poglądowości przykładowej przyjmiemy upraszczające założenie, że różne załączki prowadzą do różnych ciągów. Wówczas spośród wszystkich  $2^\ell$  możliwych ciągów,  $2^k$  występuje z prawdopodobieństwem  $1/2^k$ , podczas gdy pozostałe  $2^\ell - 2^k$  ciągów nie wystąpią nigdy. W tym przypadku rozkład prawdopodobieństwa  $p_1$  jest wysoce niejednostajny).

Choć oba rozkłady prawdopodobieństwa  $p_0$  i  $p_1$  mogą bardzo się różnić, nie jest wykluczone, że będą one  $\epsilon$ -rozróżnialne tylko dla bardzo małych wartości  $\epsilon$ . To jest nasz cel przy konstrukcji generatora bitów pseudolosowych.

### Przykład 12.3

Założymy, że PRBG generuje jedynie takie ciągi, w których dokładnie  $\ell/2$  bitów ma wartość 0, pozostałych  $\ell/2$  bitów ma zaś wartość 1. Określmy funkcję  $\mathbf{A}$  w następujący sposób:

$$\mathbf{A}(z_1, \dots, z_\ell) = \begin{cases} 1, & \text{gdy } (z_1, \dots, z_\ell) \text{ ma } \ell/2 \text{ bitów równych } 0, \\ 0, & \text{w przeciwnym przypadku.} \end{cases}$$

Algorytm  $\mathbf{A}$  jest deterministyczny. Nietrudno zauważyć, że

$$E_{\mathbf{A}}(p_0) = \frac{\binom{\ell}{\ell/2}}{2^\ell}$$

oraz

$$E_{\mathbf{A}}(p_1) = 1.$$

Można wykazać, że

$$\lim_{\ell \rightarrow \infty} \frac{\binom{\ell}{\ell/2}}{2^\ell} = 0.$$

Tak więc dla dowolnej ustalonej wartości  $\epsilon < 1$  rozkłady  $p_0$  i  $p_1$  są  $\epsilon$ -rozróżnialne dla dostatecznie dużego  $\ell$ .  $\square$

### 12.2.1. Algorytm przewidywania następnego bitu

Innym instrumentem przydatnym przy badaniu generatorów bitów pseudolosowych jest algorytm przewidywania następnego bitu. Działa on w następujący sposób. Niech  $f$  będzie generatorem bitów losowych o parametrach  $k$  i  $\ell$  ( $(k, \ell)$ -PRBG). Założymy, że mamy probabilistyczny algorytm  $\mathbf{B}_i$ , który przyjmuje na wejściu pierwszych  $i - 1$  bitów wygenerowanych przez  $f$  (przy nieznanym założku), powiedzmy,  $z_1, \dots, z_{i-1}$ , i stara się przewidzieć następny bit  $z_i$ . Wartość  $i$  ma jedynie spełniać warunek  $0 \leq i \leq \ell - 1$ . Mówimy, że  $\mathbf{B}_i$  jest *algorytmem przewidywania następnego bitu* z wiarygodnością  $\epsilon$  (lub  $\epsilon$ -algorytmem przewidywania następnego bitu), gdy  $\mathbf{B}_i$  potrafi przewidzieć  $i$ -ty bit pseudolosowego ciągu z prawdopodobieństwem równym co najmniej  $1/2 + \epsilon$ , gdzie  $\epsilon > 0$ .

Pojęcie rozkładu prawdopodobieństwa pozwoli uzyskać bardziej ścisłe sformułowanie. Zdefiniowaliśmy już rozkład prawdopodobieństwa  $p_1$  na  $(\mathbb{Z}_2)^\ell$  indukowany przez generator bitów pseudolosowych  $f$ . Możemy też rozważać rozkłady prawdopodobieństwa indukowane przez  $f$  na każdym z  $\ell$  pseudolosowych bitów wyjściowych (lub na dowolnym podzbiorze tych bitów). I tak, jeśli  $1 \leq i \leq \ell$ , to możemy myśleć o  $i$ -tym pseudolosowym bicie wyjściowym jako o zmiennej losowej, którą oznaczmy  $\mathbf{z}_i$ .

Przyjęte definicje pozwalają scharakteryzować algorytm przewidywania następnego bitu w następujący sposób.

#### TWIERDZENIE 12.1

Niech  $f$  będzie  $(k, \ell)$ -PRBG. Wówczas algorytm probabilistyczny  $\mathbf{B}_i$  jest  $\epsilon$ -algorytmem przewidywania następnego bitu wtedy i tylko wtedy, gdy

$$\sum_{(z_1, \dots, z_{i-1}) \in (\mathbb{Z}_2)^{i-1}} p_1(z_1, \dots, z_{i-1}) \cdot p(\mathbf{z}_i = \mathbf{B}_i | (z_1, \dots, z_{i-1})) \geq \frac{1}{2} + \epsilon.$$

**DOWÓD** Prawdopodobieństwo poprawnego przewidzenia  $i$ -tego bitu jest sumą (po wszystkich ciągach  $(i-1)$ -wyrazowych  $(z_1, \dots, z_{i-1})$ ) iloczynu prawdopodobieństwa tego, że ciąg  $(z_1, \dots, z_{i-1})$  został wygenerowany za pomocą generatora o funkcji  $f$  oraz prawdopodobieństwa poprawnego przewidzenia  $i$ -tego bitu na podstawie danego ciągu  $(i-1)$ -wyrazowego  $(z_1, \dots, z_{i-1})$ . ■

Wartość  $1/2 + \epsilon$  pojawia się w definicji dlatego, że każdy algorytm przewidywania następnego bitu potrafi przewidzieć następny bit z prawdopodobieństwem  $1/2$ . Jeśli ciąg nie jest losowy, to prawdopodobieństwo może być jeszcze większe. Zauważmy, że nie ma potrzeby rozpatrywania algorytmów przewidujących następny bit z prawdopodobieństwem mniejszym od  $1/2$ , gdyż w takim przypadku algorytm podający zawsze  $1 - z$  zamiast  $z$  będzie przewidywał następny bit z prawdopodobieństwem większym od  $1/2$ .

W celu zilustrowania omówionych wyżej pojęć zbudujemy algorytm przewidywania następnego bitu dla liniowego generatora kongruencyjnego z przykładu 12.1.

Przykład 12.1 cd.

Dla dowolnego  $i$ , takiego że  $1 \leq i \leq 9$ , niech  $\mathbf{B}_i(z) = 1 - z$ . Oznacza to, że  $\mathbf{B}_i$  przewiduje, że najbardziej prawdopodobnym bitem po 0 jest 1 i na odwrót. Nietrudno obliczyć na podstawie tablicy 12.1, że każdy z tych algorytmów  $\mathbf{B}_i$  jest  $\frac{9}{26}$ -algorytmem przewidywania następnego bitu, czyli przewiduje poprawnie następny bit z prawdopodobieństwem 20/31.  $\square$

Algorytm przewidywania następnego bitu może posłużyć do konstrukcji algorytmu rozróżniającego  $\mathbf{A}$ , jak to widać na rysunku 12.3. Daną wejściową dla  $\mathbf{A}$  jest ciąg bitów  $z_1, \dots, z_\ell$ , a algorytm  $\mathbf{A}$  zawiera  $\mathbf{B}_i$  jako podprogram.

Dane wejściowe: ciąg  $\ell$ -wyrazowy  $(z_1, \dots, z_\ell)$

- (1) oblicz  $z := \mathbf{B}_i(z_1, \dots, z_{i-1})$
- (2) **if**  $z = z_i$  **then**  
 $\quad \mathbf{A}(z_1, \dots, z_\ell) = 1$   
**else**  
 $\quad \mathbf{A}(z_1, \dots, z_\ell) = 0$

RYSUNEK 12.3. Konstrukcja algorytmu rozróżniającego z algorytmu przewidywania następnego bitu

### TWIERDZENIE 12.2

Niech  $\mathbf{B}_i$  będzie  $\epsilon$ -algorytmem przewidywania następnego bitu dla  $(k, \ell)$ -PRBG  $f$ . Jeśli  $p_1$  jest rozkładem prawdopodobieństwa indukowanym przez  $f$  na  $(\mathbb{Z}_2)^\ell$ , a  $p_0$  jest jednostajnym rozkładem prawdopodobieństwa na  $(\mathbb{Z}_2)^\ell$ , to algorytm  $\mathbf{A}$ , opisany na rysunku 12.3, jest  $\epsilon$ -algorytmem rozróżniającym  $p_1$  i  $p_0$ .

**DOWÓD** Zauważmy po pierwsze, że

$$\mathbf{A}(z_1, \dots, z_\ell) = 1 \Leftrightarrow \mathbf{B}_i(z_1, \dots, z_{i-1}) = z_i.$$

Wynik podawany przez  $\mathbf{A}$  nie zależy od wartości  $z_{i+1}, \dots, z_\ell$ . Możemy zatem przeprowadzić obliczenia w następujący sposób:

$$\begin{aligned} E_{\mathbf{A}}(p_1) &= \sum_{(z_1, \dots, z_\ell) \in (\mathbb{Z}_2)^\ell} p_1(z_1, \dots, z_\ell) \cdot p(\mathbf{A} = 1 | (z_1, \dots, z_\ell)) \\ &= \sum_{(z_1, \dots, z_i) \in (\mathbb{Z}_2)^i} p_1(z_1, \dots, z_i) \cdot p(\mathbf{A} = 1 | (z_1, \dots, z_i)) \\ &= \sum_{(z_1, \dots, z_i) \in (\mathbb{Z}_2)^i} p_1(z_1, \dots, z_i) \cdot p(\mathbf{B}_i = z_i | (z_1, \dots, z_i)) \\ &= \sum_{(z_1, \dots, z_{i-1}) \in (\mathbb{Z}_2)^{i-1}} p_1(z_1, \dots, z_{i-1}) \cdot p(\mathbf{B}_i = z_i | (z_1, \dots, z_{i-1})) \\ &\equiv \frac{1}{2} + \epsilon. \end{aligned}$$

Jednak każdy algorytm przewidywania  $\mathbf{B}_i$  przewidzi  $i$ -ty bit naprawdę losowego ciągu z prawdopodobieństwem  $1/2$ . Nietrudno zatem zauważyć, że  $E_{\mathbf{A}}(p_0) = 1/2$  i w rezultacie  $|E_{\mathbf{A}}(p_0) - E_{\mathbf{A}}(p_1)| \geq \epsilon$ , zgodnie z tezą. ■

Jeden z głównych wyników teorii generatorów bitów pseudolosowych to pochodzące od Yao twierdzenie, że algorytm przewidywania następnego bitu jest testem *uniwersalnym*. Oznacza to, że generator bitów pseudolosowych jest „bezpieczny” wtedy i tylko wtedy, gdy nie istnieje  $\epsilon$ -algorytm przewidywania następnego bitu, z wyjątkiem bardzo małych wartości  $\epsilon$ . Z twierdzenia 12.2 wynika implikacja w jedną stronę. Aby udowodnić brakującą implikację, musimy pokazać, jak istnienie algorytmu rozróżniającego pociąga za sobą istnienie algorytmu przewidywania następnego bitu. O tym mówi twierdzenie 12.3.

### TWIERDZENIE 12.3

Niech  $\mathbf{A}$  będzie  $\epsilon$ -algorytmem rozróżniającym  $p_1$  i  $p_0$ , gdzie  $p_1$  jest rozkładem prawdopodobieństwa na  $(\mathbb{Z}_2)^\ell$  indukowanym za pomocą generatora  $(k, \ell)$ -PRBG o funkcji  $f$ , a  $p_0$  jest jednostajnym rozkładem prawdopodobieństwa na  $(\mathbb{Z}_2)^\ell$ . Wówczas dla pewnego  $i$ ,  $1 \leq i \leq \ell - 1$ , istnieje  $\epsilon/\ell$ -algorytm przewidywania następnego bitu  $\mathbf{B}_i$  dla  $f$ .

**DOWÓD** Określmy dla  $1 \leq i \leq \ell$  rozkład prawdopodobieństwa  $q_i$  na  $(\mathbb{Z}_2)^\ell$ , gdzie pierwszych  $i$  bitów zostało uzyskanych za pomocą generatora bitów pseudolosowych o funkcji  $f$ , a pozostałych  $\ell - i$  bitów wygenerowano losowo. Wtedy  $q_0 = p_0$  i  $q_\ell = p_\ell$ . Z założenia

$$|E_{\mathbf{A}}(q_0) - E_{\mathbf{A}}(q_\ell)| \geq \epsilon.$$

Z nierówności trójkąta mamy:

$$|E_{\mathbf{A}}(q_0) - E_{\mathbf{A}}(q_\ell)| \leq \sum_{i=1}^{\ell} |E_{\mathbf{A}}(q_{i-1}) - E_{\mathbf{A}}(q_i)|.$$

Wynika stąd, że istnieje co najmniej jedna wartość  $i$ ,  $1 \leq i \leq \ell$ , dla której

$$|E_{\mathbf{A}}(q_{i-1}) - E_{\mathbf{A}}(q_i)| \geq \frac{\epsilon}{\ell}.$$

Bez utraty ogólności możemy przyjąć, że

$$E_{\mathbf{A}}(q_{i-1}) - E_{\mathbf{A}}(q_i) \geq \frac{\epsilon}{\ell}.$$

Zbudujemy algorytm przewidywania  $i$ -tego bitu (dla tej określonej wartości  $i$ ). Algorytm ten ma charakter probabilistyczny; przedstawiamy go na rysunku 12.4. Oto zasadniczy pomysł, na którym opiera się konstrukcja. Algorytm przewidywania w istocie produkuje  $\ell$ -elementowy ciąg bitów zgodnie z rozkładem prawdopodobieństwa  $q_{i-1}$ , przy założeniu że bity  $z_1, \dots, z_{i-1}$  pochodzą z generatora bitów pseudolosowych. Gdy algorytm  $\mathbf{A}$  daje odpowiedź „0”, oznacza to,

Dane wejściowe: ciąg  $(i - 1)$ -wyrazowy  $(z_1, \dots, z_{i-1})$

- (1) wybierz losowo  $(z_1, \dots, z_\ell) \in (\mathbb{Z}_2)^{\ell-i+1}$
- (2) oblicz  $z := \mathbf{A}(z_1, \dots, z_\ell)$
- (3) zdefiniuj  $\mathbf{B}_i(z_1, \dots, z_{i-1}) = (z + z_i) \bmod 2$

**RYSUNEK 12.4. Konstrukcja algorytmu przewidywania następnego bitu z algorytmu rozróżniającego**

że w jego ocenie najbardziej prawdopodobne jest to, że ciąg  $\ell$ -elementowy powstał zgodnie z rozkładem prawdopodobieństwa  $q_i$ . Rozkłady  $q_{i-1}$  i  $q_i$  różnią się jedynie tym, że w  $q_{i-1}$  bit o numerze  $i$  powstaje losowo, podczas gdy w  $q_i$  jest generowany za pomocą PRBG. Dlatego odpowiedź „0” oznacza uznanie, że  $i$ -ty bit,  $z_i$ , jest tym, co wpisałby w tym miejscu generator bitów pseudolosowych. Możemy zatem przyjąć, że  $z_i$  jest wynikiem przewidywania  $i$ -tego bitu. Gdy algorytm  $\mathbf{A}$  odpowiada „1”, uzna je tym samym, że  $z_i$  jest bitem losowym, zatem za wynik przewidywania  $i$ -tego bitu należy uznać  $1 - z_i$ .

Musimy obliczyć prawdopodobieństwo tego, że  $i$ -ty bit zostanie przewidziany poprawnie. Zauważmy, że gdy odpowiedzią jest „0”, przewidywanie jest poprawne z prawdopodobieństwem

$$p_1(z_i | (z_1, \dots, z_{i-1})),$$

gdzie  $p_1$  jest rozkładem prawdopodobieństwa indukowanym przez PRBG. Gdy  $\mathbf{A}$  odpowiada „1”, prawdopodobieństwo poprawnego przewidywania jest równe

$$1 - p_1(z_i | (z_1, \dots, z_{i-1})).$$

Przyjmijmy, dla uproszczenia zapisu,  $\mathbf{z} = (z_1, \dots, z_\ell)$ . Wykorzystamy w obliczeniach następujący fakt:

$$q_{i-1}(\mathbf{z}) \cdot p_1(z_i | (z_1, \dots, z_{i-1})) = \frac{q_i(\mathbf{z})}{2}.$$

Oto jego prosty dowód:

$$\begin{aligned} & q_{i-1}(z_1, \dots, z_\ell) \cdot p_1(z_i | (z_1, \dots, z_{i-1})) \\ &= q_{i-1}(z_1, \dots, z_{i-1}) \cdot \frac{1}{2^{\ell-i+1}} \cdot p_1(z_i | (z_1, \dots, z_{i-1})) \\ &= q_i(z_1, \dots, z_i) \cdot \frac{1}{2^{\ell-i+1}} \\ &= \frac{q_i(z_1, \dots, z_\ell)}{2}. \end{aligned}$$

Możemy teraz przystąpić do głównych obliczeń.

$$\begin{aligned}
 & p(\mathbf{z}_i = \mathbf{B}_i(z_1, \dots, z_{i-1})) \\
 &= \sum_{\mathbf{z} \in (\mathbb{Z}_2)^\ell} q_{i-1}(\mathbf{z}) [p(\mathbf{A} = 0|\mathbf{z}) \cdot p_1(z_i|(z_1, \dots, z_{i-1})) \\
 &\quad + p(\mathbf{A} = 1|\mathbf{z}) \cdot (1 - p_1(z_i|(z_1, \dots, z_{i-1})))] \\
 &= \sum_{\mathbf{z} \in (\mathbb{Z}_2)^\ell} \frac{q_i(\mathbf{z})}{2} \cdot p(\mathbf{A} = 0|\mathbf{z}) + \sum_{\mathbf{z} \in (\mathbb{Z}_2)^\ell} q_{i-1}(\mathbf{z}) \cdot p(\mathbf{A} = 1|\mathbf{z}) \\
 &\quad - \sum_{\mathbf{z} \in (\mathbb{Z}_2)^\ell} \frac{q_i(\mathbf{z})}{2} \cdot p(\mathbf{A} = 1|\mathbf{z}) \\
 &= \frac{1 - E_{\mathbf{A}}(q_i)}{2} + E_{\mathbf{A}}(q_{i-1}) - \frac{E_{\mathbf{A}}(q_i)}{2} \\
 &= \frac{1}{2} + E_{\mathbf{A}}(q_{i-1}) - E_{\mathbf{A}}(q_i) \\
 &\geq \frac{1}{2} + \frac{\epsilon}{\ell},
 \end{aligned}$$

co należało udowodnić. ■

### 12.3. Generator Bluma–Bluma–Shuba

Opiszymy tu jeden z najczęściej stosowanych generatorów bitów pseudolosowych, którego pomysłodawcami byli Blum, Blum i Shub. Zaczniemy od przypomnienia kilku wyników dotyczących symboli Jacobiego z podrozdziału 4.5 oraz paru faktów teorioliczbowych z innych części rozdziału 4.

Niech  $p$  i  $q$  będą dwiema różnymi liczbami pierwszymi i niech  $n = pq$ . Przyпомнijmy symbol Jacobiego:

$$\left( \frac{x}{n} \right) = \begin{cases} 0, & \text{gdy NWD}(x, n) > 1, \\ 1, & \text{gdy } \left( \frac{x}{p} \right) = \left( \frac{x}{q} \right) = 1 \text{ lub } \left( \frac{x}{p} \right) = \left( \frac{x}{q} \right) = -1, \\ -1, & \text{gdy jedna z liczb } \left( \frac{x}{p} \right) \text{ i } \left( \frac{x}{q} \right) \text{ jest równa 1, a druga } -1. \end{cases}$$

Niech  $\text{QR}(n)$  oznacza zbiór reszt kwadratowych modulo  $n$ :

$$\text{QR}(n) = \{x^2 \bmod n : x \in \mathbb{Z}_n^*\}.$$

Przypomnijmy, że  $x$  jest resztą kwadratową modulo  $n$  wtedy i tylko wtedy, gdy

$$\left( \frac{x}{p} \right) = \left( \frac{x}{q} \right) = 1.$$

Określmy

$$\widetilde{\text{QR}}(n) = \left\{ x \in \mathbb{Z}_n^* \setminus \text{QR}(n) : \left( \frac{x}{n} \right) = 1 \right\}.$$

Wtedy

$$\widetilde{\text{QR}}(n) = \left\{ x \in \mathbb{Z}_n^* : \left( \frac{x}{p} \right) = \left( \frac{x}{q} \right) = 1 \right\}.$$

Elementy zbioru  $\widetilde{\text{QR}}(n)$  nazywamy *pseudokwadratami* modulo  $n$ .

**Dane** Dodatnia liczba całkowita  $n$ , będąca iloczynem dwóch nieznanego liczb pierwszych  $p$  i  $q$ , oraz liczba całkowita  $x \in \mathbb{Z}_n^*$ , taka że  $\left( \frac{x}{n} \right) = 1$ .

**Pytanie** Czy  $x$  jest resztą kwadratową modulo  $n$ ?

#### RYSUNEK 12.5. Reszty kwadratowe

Podstawą **generatora Bluma–Bluma–Shuba** (w skrócie generatora BBS), podobnie jak wielu innych systemów kryptograficznych, jest **problem reszt kwadratowych** opisany na rysunku 12.5. (W rozdziale 4 zdefiniowaliśmy reszty kwadratowe modulo liczba pierwsza i wykazaliśmy, że taki problem ma łatwe rozwiązanie; tu mamy jednak do czynienia z modelem złożonym). Zauważmy, że problem reszt kwadratowych wymaga odróżnienia reszt kwadratowych modulo  $n$  od pseudokwadratów modulo  $n$ . Można uznać, że nie jest to trudniejsze od rozkładu liczby  $n$  na czynniki pierwsze. Rzeczywiście, jeśli można obliczyć faktoryzację  $n = pq$ , to bez trudu można uzyskać wartość, na przykład,  $\left( \frac{x}{p} \right)$ .

Jeśli  $\left( \frac{x}{n} \right) = 1$ , to  $x$  jest resztą kwadratową wtedy i tylko wtedy, gdy  $\left( \frac{x}{p} \right) = 1$ .

Nie widać natomiast żadnego sposobu na rozwiązanie problemu reszt kwadratowych, gdy nie jest znany rozkład na czynniki pierwsze liczby  $n$ . Problem ten wydaje się więc efektywnie nierozwiązalny, jeśli efektywnie nierozwiązalny jest rozkład na czynniki pierwsze liczby  $n$ .

Generator Bluma–Bluma–Shuba przedstawiamy na rysunku 12.6. Jego działanie jest dość proste. Dla danego założka  $s_0 \in \text{QR}(n)$  obliczamy ciąg  $s_1, s_2, \dots, s_\ell$ , biorąc kolejno kwadraty modulo  $n$ , po czym redukujemy każdą wartość  $s_i$  modulo 2. Jeśli tak otrzymaną na  $i$ -tym miejscu wartość nazwiemy  $z_i$ , to

$$z_i = \left( s_0^{2^i} \bmod n \right) \bmod 2,$$

$$1 \leq i \leq \ell.$$

Podamy teraz przykład generatora BBS.

Niech  $p, q$  będą dwiema  $(k/2)$ -bitowymi liczbami pierwszymi, takimi że  $p \equiv q \equiv 3 \pmod{4}$  i niech  $n = pq$ .  $\text{QR}(n)$  oznacza zbiór reszt kwadratowych modulo  $n$ .

Załązek  $s_0$  jest dowolnym elementem zbioru  $\text{QR}(n)$ . Dla  $i \geq 0$  definiujemy

$$s_{i+1} = s_i^2 \pmod{n},$$

a wówczas

$$f(s_0) = (z_1, z_2, \dots, z_\ell),$$

gdzie

$$z_i = s_i \pmod{2},$$

$1 \leq i \leq \ell$ . Wtedy  $f$  jest  $(k, \ell)$ -PRBG, zwany generatorem Bluma–Bluma–Shuba, w skrócie generatorem BBS.

### RYSUNEK 12.6. Generator Bluma–Bluma–Shuba

#### Przykład 12.4

Niech  $n = 192649 = 383 \cdot 503$  i  $s_0 = 101355^2 \pmod{n} = 20749$ . Pierwsze 20 bitów wygenerowanych za pomocą generatora BBS zawiera tablica 12.3. Jak widać, załóżek generuje następujący ciąg bitów:

11001110000100111010.

□

Zatrzymajmy się teraz nad pewną cechą generatora BBS, istotną dla jego bezpieczeństwa. Z równości  $n = pq$ , gdzie  $p \equiv q \equiv 3 \pmod{4}$  wynika, że dla każdej reszty kwadratowej  $x$  istnieje dokładnie jeden pierwiastek kwadratowy z  $x$ , który jest także resztą kwadratową. Taki pierwiastek nazwiemy *głównym* pierwiastkiem kwadratowym z  $x$ . Przekształcenie  $x \mapsto x^2 \pmod{n}$  stosowane w konstrukcji generatora BBS jest więc permutacją na zbiorze  $\text{QR}(n)$  reszt kwadratowych modulo  $n$ .

#### 12.3.1. Bezpieczeństwo generatora BBS

W tym punkcie zajmiemy się szczegółowo bezpieczeństwem generatora BBS. Zaczniemy od przyjęcia założenia, że pseudolosowe bity uzyskiwane za pomocą generatora BBS są  $\epsilon$ -odróżnialne od  $\ell$  losowych bitów, i zobaczymy, dokąd nas to założenie doprowadzi. Zakładamy tu, że  $n = pq$ , gdzie  $p$  i  $q$  są liczbami pierwszymi, takimi że  $p \equiv q \equiv 3 \pmod{4}$ , oraz że nie jest znany rozkład na czynniki pierwsze  $n = pq$ .

TABLICA 12.3. Bity wytworzone za pomocą generatora BBS

| $i$ | $s_i$  | $z_i$ |
|-----|--------|-------|
| 0   | 20749  |       |
| 1   | 143135 | 1     |
| 2   | 177671 | 1     |
| 3   | 97048  | 0     |
| 4   | 89992  | 0     |
| 5   | 174051 | 1     |
| 6   | 80649  | 1     |
| 7   | 45663  | 1     |
| 8   | 69442  | 0     |
| 9   | 186894 | 0     |
| 10  | 177046 | 0     |
| 11  | 137922 | 0     |
| 12  | 123175 | 1     |
| 13  | 8630   | 0     |
| 14  | 114386 | 0     |
| 15  | 14863  | 1     |
| 16  | 133015 | 1     |
| 17  | 106065 | 1     |
| 18  | 45870  | 0     |
| 19  | 137171 | 1     |
| 20  | 48060  | 0     |

Poznaliśmy już pojęcie algorytmu przewidywania następnego bitu. Tu rozważymy podobne pojęcie, które nazwiemy *algorytmem przewidywania poprzedniego bitu*. Taki algorytm dla  $(k, \ell)$ -generatora BBS pobiera  $\ell$  pseudolosowych bitów wyprodukowanych przez generator (na podstawie nieznanego załączka  $s_0 \in \text{QR}(n)$ ), po czym próbuje przewidzieć wartość  $z_0 = s_0 \bmod 2$ . Może to być algorytm probabilistyczny. Mówimy, że algorytm przewidywania poprzedniego bitu  $\mathbf{B}_0$  jest  $\epsilon$ -algorytmem, jeśli prawdopodobieństwo poprawnego wskazania  $z_0$  jest równe co najmniej  $1/2 + \epsilon$ , gdzie prawdopodobieństwo obliczamy po wszystkich możliwych załączkach  $s_0$ .

Podajemy bez dowodu twierdzenie podobne do twierdzenia 12.3.

#### TWIERDZENIE 12.4

Niech  $\mathbf{A}$  będzie  $\epsilon$ -algorytmem rozróżniającym  $p_1$  i  $p_0$ , gdzie  $p_1$  jest rozkładem prawdopodobieństwa na  $(\mathbb{Z}_2)^\ell$  indukowanym przez  $(k, \ell)$ -generator BBS o funkcji  $f$ , a  $p_0$  jest jednostajnym rozkładem prawdopodobieństwa na  $(\mathbb{Z}_2)^\ell$ . Wówczas istnieje  $\epsilon/\ell$ -algorytm przewidywania poprzedniego bitu  $\mathbf{B}_0$  dla  $f$ .

Pokażemy teraz, jak wykorzystać  $\epsilon/\ell$ -algorytm przewidywania poprzedniego bitu  $\mathbf{B}_0$  do konstrukcji probabilistycznego algorytmu odróżniającego reszty kwadratowe modulo  $n$  od pseudoreszt kwadratowych modulo  $n$  z prawdopodobieństwem  $1/2 + \epsilon$ . Ten algorytm  $\mathbf{A}$ , przedstawiony na rysunku 12.7, używa  $\mathbf{B}_0$  jako podprogramu (lub wyroczni).

Dane wejściowe:  $x \in \mathbb{Z}_n^*$ , taka że  $(\frac{x}{n}) = 1$

- (1) oblicz  $s_0 = x^2 \bmod n$  oraz  $z_0 = s_0 \bmod 2$
- (2) startując od  $s_0$ , oblicz  $z_1, \dots, z_{\ell-1}$  za pomocą generatora BBS
- (3) oblicz  $z = \mathbf{B}_0(z_0, \dots, z_{\ell-1})$
- (4) **if**  $(x \bmod 2) = z$  **then**  
    odpowiedź „ $x \in \text{QR}(n)$ ”  
**else**  
    odpowiedź „ $x \in \widetilde{\text{QR}}(n)$ ”

RYSUNEK 12.7. Konstrukcja algorytmu rozróżniającego reszty kwadratowe z algorytmu przewidywania poprzedniego bitu

### TWIERDZENIE 12.5

Niech  $\mathbf{B}_0$  będzie  $\epsilon$ -algorytmem przewidywania poprzedniego bitu dla  $(k, \ell)$ -generatora BBS o funkcji  $f$ . Wówczas algorytm  $\mathbf{A}$ , przedstawiony na rysunku 12.7, poprawnie wskazuje reszty kwadratowe z prawdopodobieństwem równym co najmniej  $1/2 + \epsilon$ , gdzie prawdopodobieństwo obliczamy względem wszystkich możliwych wartości wejściowych  $x \in \text{QR}(n) \cup \widetilde{\text{QR}}(n)$ .

**DOWÓD** Z warunków  $n = pq$  i  $p \equiv q \equiv 3 \pmod{4}$  wynika, że  $(\frac{-1}{n}) = 1$ , zatem  $-1 \in \widetilde{\text{QR}}(n)$ . Stąd, jeśli  $(\frac{x}{n}) = 1$ , to główny pierwiastek kwadratowy liczby  $s_0 = x^2$  jest równy  $x$ , gdy  $x \in \text{QR}(n)$ , lub  $-x$ , gdy  $x \in \widetilde{\text{QR}}(n)$ . Jednak

$$(-x \bmod n) \bmod 2 \neq (x \bmod n) \bmod 2,$$

a więc algorytm  $\mathbf{A}$  podaje poprawną odpowiedź wtedy i tylko wtedy, gdy  $\mathbf{B}_0$  poprawnie przewiduje wartość  $z$ . Stąd już natychmiast wynika teza twierdzenia. ■

Twierdzenie 12.5 pokazuje, jak można odróżnić pseudokwadraty od reszt kwadratowych z prawdopodobieństwem równym co najmniej  $1/2 + \epsilon$ . Wykażemy teraz, że wynik ten prowadzi do algorytmu typu Monte Carlo, podającego poprawną odpowiedź z prawdopodobieństwem równym co najmniej  $1/2 + \epsilon$ . Innymi słowy, dla dowolnego  $x \in \text{QR}(n) \cup \widetilde{\text{QR}}(n)$  algorytm typu Monte Carlo daje poprawną odpowiedź z prawdopodobieństwem równym co najmniej  $1/2 + \epsilon$ . Zwrócić uwagę na to, że jest to algorytm *neutralny* (odpowiedź nieprawidłowa może pojawić się po dowolnych danych wejściowych) w przeciwieństwie do algorytmów Monte Carlo rozważanych w podrozdziale 4.5, nastawionych pozytywnie lub negatywnie.

Algorytm Monte Carlo  $\mathbf{A}_1$  przedstawiamy na rysunku 12.8. Używa on poprzedniego algorytmu  $\mathbf{A}$  jako podprogramu.

### TWIERDZENIE 12.6

Jeśli algorytm  $\mathbf{A}$  poprawnie wskazuje na reszty kwadratowe z prawdopodobieństwem równym co najmniej  $1/2 + \epsilon$ , to algorytm  $\mathbf{A}_1$  opisany na rysunku 12.8, jest

Dane wejściowe:  $x \in \mathbb{Z}_n^*$ , taka że  $(\frac{x}{n}) = 1$

- (1) wybierz losowo  $r \in \mathbb{Z}_n^*$
- (2) z prawdopodobieństwem  $1/2$ , oblicz  
 $x' = r^2 x \bmod n$ ,  
 w przeciwnym razie oblicz  
 $x' = -r^2 x \bmod n$
- (3) wywołaj  $\mathbf{A}(x')$ , uzyskując odpowiedź „QR” lub „ $\widetilde{\text{QR}}$ ”
- (4) **if**  
 $\mathbf{A}(x') = \text{QR}$  oraz  $x' = r^2 x \bmod n$   
**or**  
 $\mathbf{A}(x') = \widetilde{\text{QR}}$  oraz  $x' = -r^2 x \bmod n$   
**then**  
 odpowiedz „ $x \in \text{QR}$ ”  
**else**  
 odpowiedz „ $x \in \widetilde{\text{QR}}$ ”

RYSUNEK 12.8. Algorytm Monte Carlo dla reszt kwadratowych

algorytmem typu Monte Carlo odróżniającym reszty kwadratowe z prawdopodobieństwem błędu równym co najwyżej  $1/2 - \epsilon$ .

**DOWÓD** Dla dowolnej liczby wejściowej  $x \in \text{QR}(n) \cup \widetilde{\text{QR}}(n)$  w kroku 2 algorytmu  $\mathbf{A}_1$  pojawia się losowy element  $x'$  ze zbioru  $\text{QR}(n) \cup \widetilde{\text{QR}}(n)$ , którego status jako reszty kwadratowej znamy. ■

Ostatni krok polega na wykazaniu, że dowolny (neutralny) algorytm Monte Carlo o prawdopodobieństwie błędu równym co najwyżej  $1/2 - \epsilon$  może posłużyć do zbudowania neutralnego algorytmu Monte Carlo z prawdopodobieństwem błędu równym co najwyżej  $\delta$ , dla dowolnej wartości  $\delta > 0$ . Inaczej mówiąc, możemy uczynić prawdopodobieństwo uzyskania poprawnej odpowiedzi dowolnie bliskim 1. Pomyśl polega na uruchomieniu danego algorytmu Monte Carlo  $2m + 1$  razy dla pewnej liczby naturalnej  $m$  i uznaniu za poprawną odpowiedź występującą najczęściej. Obliczając prawdopodobieństwo błędu tego algorytmu, widzimy, jak  $m$  zależy od wybranej wartości  $\delta$ . O tej zależności mówi następujące twierdzenie.

### TWIERDZENIE 12.7

Niech  $\mathbf{A}_1$  będzie neutralnym algorytmem typu Monte Carlo z prawdopodobieństwem błędu równym co najwyżej  $1/2 - \epsilon$ . Jeśli algorytm  $\mathbf{A}_1$  uruchamiamy  $n = 2m + 1$  razy dla tych samych danych wejściowych  $I$  i wybieramy najczęściej występującą odpowiedź, to prawdopodobieństwo błędu dla takiego algorytmu nie przekracza liczby

$$\frac{(1 - 4\epsilon^2)^m}{2}.$$

**DOWÓD** Prawdopodobieństwo uzyskania dokładnie  $i$  poprawnych odpowiedzi w  $n$  próbach jest równe co najwyżej

$$\binom{n}{i} \left(\frac{1}{2} + \epsilon\right)^i \left(\frac{1}{2} - \epsilon\right)^{n-i}.$$

Prawdopodobieństwo tego, że najczęstsza odpowiedź jest błędna, jest równe prawdopodobieństwu tego, że liczba poprawnych odpowiedzi w  $n$  próbach jest równa co najwyżej  $m$ . Mamy zatem:

$$\begin{aligned} p(\text{błąd}) &\leq \sum_{i=0}^m \binom{n}{i} \left(\frac{1}{2} + \epsilon\right)^i \left(\frac{1}{2} - \epsilon\right)^{2m+1-i} \\ &= \left(\frac{1}{2} + \epsilon\right)^m \left(\frac{1}{2} - \epsilon\right)^{m+1} \sum_{i=0}^m \binom{n}{i} \left(\frac{1/2 - \epsilon}{1/2 + \epsilon}\right)^{m-i} \\ &\leq \left(\frac{1}{2} + \epsilon\right)^m \left(\frac{1}{2} - \epsilon\right)^{m+1} \sum_{i=0}^m \binom{n}{i} \\ &= \left(\frac{1}{2} + \epsilon\right)^m \left(\frac{1}{2} - \epsilon\right)^{m+1} 2^{2m} \\ &= \left(\frac{1}{4} - \epsilon^2\right)^m \left(\frac{1}{2} - \epsilon\right) 2^{2m} \\ &= (1 - 4\epsilon^2)^m \left(\frac{1}{2} - \epsilon\right) \\ &\leq \frac{(1 - 4\epsilon^2)^m}{2}, \end{aligned}$$

co należało wykazać. ■

Przypuśćmy, że chcemy zmniejszyć prawdopodobieństwo błędu do pewnej wartości  $\delta$ , takiej że  $0 < \delta < 1/2 - \epsilon$ . Musimy zatem wybrać taką wartość  $m$ , aby był spełniony warunek

$$\frac{(1 - 4\epsilon^2)^m}{2} \leq \delta.$$

Wystarczy zatem przyjąć

$$m = \left\lceil \frac{1 + \log_2 \delta}{\log_2(1 - 4\epsilon^2)} \right\rceil.$$

Wówczas, jeśli uruchomimy algorytm **A**  $2m + 1$  razy, odpowiedź występująca najczęściej będzie poprawna z prawdopodobieństwem równym co najmniej  $1 - \delta$ . Nietrudno wykazać, że odpowiednia wartość  $m$  nie przekracza  $c/(\delta\epsilon^2)$  dla pewnej stałej  $c$ . Tak więc liczba koniecznych przebiegów algorytmu zależy wielomianowo od  $1/\delta$  i  $1/\epsilon$ .

### Przykład 12.5

Założymy, że startujemy z algorytmem Monte Carlo podającym poprawną odpowiedź z prawdopodobieństwem równym co najmniej 0,55, a więc  $\epsilon = 0,05$ . Jeśli chcemy uzyskać algorytm Monte Carlo z prawdopodobieństwem błędu nieprzekraczającym 0,05, wystarczy przyjąć  $m = 230$  i  $n = 461$ .  $\square$

Podsumujmy wszystkie redukcje, które udało nam się uzyskać do tej pory. Mamy następujący ciąg implikacji:

$(k, \ell)$ -generator BBS jest  $\epsilon$ -odróżnialny od losowego ciągu  $\ell$  bitów



$\epsilon/\ell$ -algorytm przewidywania poprzedniego bitu dla  $(k, \ell)$ -generatora BBS



algorytm rozróżniający dla reszt kwadratowych poprawny z prawdopodobieństwem równym co najmniej  $1/2 + \epsilon/\ell$



neutralny algorytm Monte Carlo dla reszt kwadratowych z prawdopodobieństwem błędu równym co najwyżej  $1/2 - \epsilon/\ell$



neutralny algorytm Monte Carlo dla reszt kwadratowych z prawdopodobieństwem błędu równym co najwyżej  $\delta$ , dla dowolnej wartości  $\delta > 0$

Ponieważ panuje powszechnie przekonanie, że nie istnieje algorytm Monte Carlo dla reszt kwadratowych z małym prawdopodobieństwem błędu, który działałby w czasie wielomianowym, możemy uznać, że generator BBS jest bezpieczny.

Kończąc ten podrozdział, wspomnijmy jeszcze o pewnym sposobie ulepszenia skuteczności generatora BBS. Ciąg bitów pseudolosowych powstaje w taki sposób, że bierze się najmniej znaczący bit z każdego wyrazu  $s_i$ , gdzie  $s_i = s_0^{2^i} \bmod n$ . Przypuśćmy, że zamiast tego z każdego  $s_i$  bierzemy  $m$  najmniej znaczących bitów. Poprawia to skuteczność generatora bitów pseudolosowych o czynnik  $m$ . Trzeba jednak sprawdzić, czy nie zagraża to bezpieczeństwu generatora. Udowodniono, że bezpieczeństwo zostaje zachowane, jeśli  $m \leq \log_2 \log_2 n$ . Możemy zatem brać około  $\log_2 \log_2 n$  bitów pseudolosowych w każdym modularnym podnoszeniu do kwadratu. W rzeczywistych implementacjach generatora BBS  $n \approx 10^{160}$ , zatem przy każdym podnoszeniu do kwadratu możemy brać 9 bitów.

## 12.4. Szyfrowanie probabilistyczne

Pomysł szyfrowania probabilistycznego pochodzi od Goldwassera i Micaliego. Popatrzmy na jeden z powodów stosowania metody. Przypuśćmy, że dysponujemy systemem kryptograficznym z kluczem publicznym i chcemy zaszyfrować jeden bit, a więc  $x = 0$  lub  $1$ . Każdy potrafi obliczyć  $e_K(0)$  i  $e_K(1)$ , zatem przeciwnik bez trudu ustaliłby, czy tekst zaszyfrowany  $y$  reprezentuje  $0$  czy  $1$ . Ogólniej, przeciwnik może zawsze ustalić, czy tekst jawnny ma określoną wartość, szyfrując hipotetyczny tekst jawnny i porównując wynik z posiadanym tekstem zaszyfrowanym.

Szyfrowanie probabilistyczne ma na celu uniemożliwienie obliczenia jakiegokolwiek informacji o tekście jawnym na podstawie znajomości tekstu zaszyfrowanego (w czasie wielomianowym). Można ten cel osiągnąć w systemach kryptograficznych z kluczem publicznym, w których szyfrowanie odbywa się w sposób probabilistyczny, a nie deterministyczny. W sytuacji kiedy jednemu tekstowi jawnemu odpowiada wiele możliwych tekstów zaszyfrowanych nie można ustalić, czy dany tekst zaszyfrowany odpowiada określzonemu tekstowi jawnemu.

Oto formalna definicja tego pojęcia.

### DEFINICJA 12.3

*Probabilistycznym systemem kryptograficznym z kluczem publicznym* nazywamy dowolną szóstkę uporządkowaną  $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{R})$ , w której  $\mathcal{P}$  jest zbiorem tekstów jawnych,  $\mathcal{C}$  jest zbiorem tekstów zaszyfrowanych,  $\mathcal{K}$  jest przestrzenią kluczy,  $\mathcal{R}$  jest zbiorem parametrów losowości, a dla każdego klucza  $K \in \mathcal{K}$ ,  $e_K \in \mathcal{E}$  jest publiczną regułą szyfrowania,  $d_K \in \mathcal{D}$  jest tajną regułą deszyfrowania. Ponadto są spełnione następujące warunki:

- (1) Dla każdego  $K \in \mathcal{K}$ ,  $e_K : \mathcal{P} \times \mathcal{R} \rightarrow \mathcal{C}$  i  $d_K : \mathcal{C} \rightarrow \mathcal{P}$  są funkcjami, takimi że

$$d_K(e_K(b, r)) = b$$

dla dowolnego tekstu jawnego  $b \in \mathcal{P}$  i każdego  $r \in \mathcal{R}$ . (W szczególności wynika stąd, że  $e_K(x, r) \neq e_K(x', r)$ , gdy  $x \neq x'$ ).

- (2) Niech  $\epsilon$  będzie ustalonym *parametrem bezpieczeństwa*. Dla dowolnego klucza  $K \in \mathcal{K}$  i  $x \in \mathcal{P}$  definiujemy rozkład prawdopodobieństwa  $p_{K,x}$  na zbiorze  $\mathcal{C}$ , przy czym  $p_{K,x}(y)$  oznacza prawdopodobieństwo tego, że  $y$  jest tekstem zaszyfrowanym, jeśli tekstem jawnym jest  $x$ , a kluczem jest  $K$  (to prawdopodobieństwo obliczamy względem wszystkich wartości  $r \in \mathcal{R}$ ). Jeśli  $x, x' \in \mathcal{P}$ ,  $x \neq x'$  oraz  $K \in \mathcal{K}$ , to rozkłady  $p_{K,x}$  i  $p_{K,x'}$  nie są  $\epsilon$ -odróżnialne.

A oto jak wygląda działanie takiego systemu. Szyfrowanie tekstu jawnego  $x$  polega na wyborze parametru losowości  $r \in \mathcal{R}$  i obliczeniu  $y = e_K(x, r)$ . Każda otrzymana w ten sposób wartość  $y = e_K(x, r)$  jest więc potencjalnym szyfrem

tekstu  $x$ . Własność 2 w definicji stwierdza, że rozkład prawdopodobieństwa wszystkich szyfrowań tekstu  $x$  nie można odróżnić od rozkładu prawdopodobieństwa wszystkich szyfrowań  $x'$ , gdy  $x \neq x'$ . Mniej formalnie, zaszyfrowany tekst  $x$  „wygląda” tak samo jak zaszyfrowany tekst  $x'$ . Parametr bezpieczeństwa  $\epsilon$  powinien być mały; w praktyce użylibyśmy wartości  $\epsilon = c/|\mathcal{R}|$  dla pewnej malej stałej  $c > 0$ .

Niech  $n = pq$ , gdzie  $p$  i  $q$  są liczbami pierwszymi, i niech  $m \in \widetilde{\text{QR}}(n)$ . Liczby całkowite  $n$  i  $m$  są publicznie znane, natomiast faktoryzacja  $n = pq$  jest tajna. Niech  $\mathcal{P} = \{0, 1\}$ ,  $\mathcal{C} = \mathcal{R} = \mathbb{Z}_n^*$ . Definiujemy

$$\mathcal{K} = \{(n, p, q, m) : n = pq, p, q \text{ pierwsze}, m \in \widetilde{\text{QR}}(n)\}.$$

Dla  $K = (n, p, q, m)$  określamy

$$e_K(x, r) = m^x r^2 \bmod n$$

oraz

$$d_K(y) = \begin{cases} 0, & \text{gdy } y \in \text{QR}(n) \\ 1, & \text{gdy } y \notin \text{QR}(n), \end{cases}$$

gdzie  $x = 0$  lub  $1$  oraz  $r, y \in \mathbb{Z}_n^*$ .

**RYSUNEK 12.9.** Probabilistyczny system kryptograficzny Goldwassera-Micaliego z kluczem publicznym

Na rysunku 12.9 jest pokazany **probabilistyczny system kryptograficzny Goldwassera–Micaliego z kluczem publicznym**. System ten szyfruje tekst bit po bicie. Szyfrem bitu 0 jest losowa reszta kwadratowa modulo  $n$ , szyfrem 1 jest losowy pseudokwadrat modulo  $n$ . Gdy Bolek dostaje element  $y \in \text{QR}(n) \cup \widetilde{\text{QR}}(n)$ , może wykorzystać swoją wiedzę o rozkładzie na czynniki pierwsze liczby  $n$  do ustalenia, czy  $y \in \text{QR}(n)$ , czy raczej  $y \in \widetilde{\text{QR}}(n)$ . W tym celu oblicza

$$\left(\frac{y}{p}\right) = (y)^{(p-1)/2} \bmod p.$$

Wtedy

$$y \in \text{QR}(n) \Leftrightarrow \left(\frac{y}{p}\right) = 1.$$

Blum i Goldwasser zaproponowali inny, bardziej skuteczny probabilistyczny system kryptograficzny z kluczem publicznym. Przedstawiamy go na rysunku 12.10. Zasadniczy pomysł **probabilistycznego systemu kryptograficznego Bluma–Goldwassera z kluczem publicznym** jest następujący.

Niech  $n = pq$ , gdzie  $p$  i  $q$  są liczbami pierwszymi,  $p \equiv q \equiv 3 \pmod{4}$ . Liczba całkowita  $n$  jest znana publicznie, natomiast jej rozkład  $n = pq$  jest tajny. Przyjmijmy  $\mathcal{P} = (\mathbb{Z}_2)^\ell$ ,  $\mathcal{C} = (\mathbb{Z}_2)^\ell \times \mathbb{Z}_n^*$ ,  $\mathcal{R} = \mathbb{Z}_n^*$  oraz

$$\mathcal{K} = \{(n, p, q) : n = pq, p, q \text{ pierwsze}\}.$$

Przyjmując  $K = (n, p, q)$  i  $r \in \mathbb{Z}_n^*$ , element  $x \in (\mathbb{Z}_2)^\ell$  szyfrujemy w sposób następujący:

- (1) za pomocą generatora BBS oblicz  $z_1, \dots, z_\ell$ , wychodząc od założka  $s_0 = r$
- (2) oblicz  $s_{\ell+1} = s_0^{2^{\ell+1}} \pmod{n}$
- (3) oblicz  $y_i = (x_i + z_i) \pmod{2}$  dla  $1 \leq i \leq \ell$
- (4) zdefiniuj  $e_K(x, r) = (y_1, \dots, y_\ell, s_{\ell+1})$

Z kolei Bolek, deszyfrując  $y$ , realizuje następujący algorytm:

- (1) oblicz  $a_1 = ((p+1)/4)^{\ell+1} \pmod{p-1}$
- (2) oblicz  $a_2 = ((q+1)/4)^{\ell+1} \pmod{q-1}$
- (3) oblicz  $b_1 = s_{\ell+1}^{a_1} \pmod{p}$
- (4) oblicz  $b_2 = s_{\ell+1}^{a_2} \pmod{q}$
- (5) znajdź, korzystając z chińskiego twierdzenia o resztach, wartość  $s_0$ , taką że

$$s_0 \equiv b_1 \pmod{p}$$

oraz

$$s_0 \equiv b_2 \pmod{q}.$$

- (6) za pomocą generatora BBS oblicz  $z_1, \dots, z_\ell$ , wychodząc od założka  $s_0 = r$
- (7) oblicz  $x_i = (y_i + z_i) \pmod{2}$  dla  $1 \leq i \leq \ell$
- (8) tekstem jawnym jest  $x = (x_1, \dots, x_\ell)$

**RYSUNEK 12.10.** Probabilistyczny system kryptograficzny Bluma-Goldwassera z kluczem publicznym

Losowy załączek  $s_0$  służy do generowania ciągu  $\ell$  pseudolosowych bitów  $z_1, \dots, z_\ell$  za pomocą generatora BBS. Wartości  $z_i$  używa się jako klucza strumieniowego, a więc oblicza się ich różnice symetryczne z kolejnymi  $\ell$  bitami tekstu jawnego; w ten sposób powstaje tekst zaszyfrowany. Także element o numerze  $\ell + 1$ , czyli  $s_{\ell+1} = s_0^{2^{\ell+1}} \pmod n$  jest przekazywany jako część tekstu zaszyfrowanego.

Gdy Bolek odbiera tekst zaszyfrowany, oblicza  $s_0$  na podstawie znajomości  $s_{\ell+1}$ , a następnie odtwarza klucz strumieniowy. Pozostaje tylko obliczyć różnice symetryczne wyrazów klucza strumieniowego z kolejnymi bitami tekstu zaszyfrowanego, by otrzymać w końcu tekst jawnego. Należy jeszcze wyjaśnić, jak odtworzyć  $s_0$ , znając  $s_{\ell+1}$ . Przypomnijmy, że każdy wyraz  $s_{i-1}$  jest głównym pierwiastkiem kwadratowym wyrazu  $s_i$ . Z założenia  $n = pq$  i  $p \equiv q \equiv 3 \pmod 4$ , zatem pierwiastki kwadratowe dowolnej reszty kwadratowej  $x$  modulo  $p$  są postraci  $\pm x^{(p+1)/4}$ . Odwołując się do własności symbolu Jacobiego, mamy

$$\begin{aligned} \left(\frac{x^{(p+1)/4}}{p}\right) &= \left(\frac{x}{p}\right)^{(p+1)/4} \\ &= 1. \end{aligned}$$

Wynika stąd, że  $x^{(p+1)/4}$  jest głównym pierwiastkiem kwadratowym z  $x$  modulo  $p$ . Podobnie,  $x^{(q+1)/4}$  jest głównym pierwiastkiem kwadratowym z  $x$  modulo  $q$ . Korzystając z chińskiego twierdzenia o resztach, możemy znaleźć główny pierwiastek kwadratowy z  $x$  modulo  $n$ .

Ogólniej,  $x^{((p+1)/4)^{\ell+1}}$  jest głównym pierwiastkiem stopnia  $2^{\ell+1}$  z  $x$  modulo  $p$ , a  $x^{((q+1)/4)^{\ell+1}}$  jest głównym pierwiastkiem stopnia  $2^{\ell+1}$  z  $x$  modulo  $q$ . Zauważysz na rząd  $\mathbb{Z}_p^*$ , równy  $p - 1$ , możemy w obliczeniu wartości  $x^{((p+1)/4)^{\ell+1}} \pmod p$  zredukować wykładnik  $((p+1)/4)^{\ell+1}$  modulo  $p - 1$ . Podobnie możemy zredukować wykładnik  $((q+1)/4)^{\ell+1}$  modulo  $q - 1$ . Na rysunku 12.10, po obliczeniu głównych pierwiastków stopnia  $2^{\ell+1}$  z  $s_{\ell+1}$  modulo  $p$  i modulo  $q$  (kroki od 1 do 4 w procesie deszyfrowania), użyto chińskiego twierdzenia o resztach, by obliczyć główny pierwiastek stopnia  $2^{\ell+1}$  z  $s_{\ell+1}$  modulo  $n$ .

Popatrzmy na przykład.

### Przykład 12.6

Niech  $n = 192649$ , tak jak w przykładzie 12.4. Przyjmijmy, że Alicja wybiera  $r = 20749$ , chcąc zaszyfrować 20-bitowy tekst jawnny

$$x = 11010011010011101101.$$

Najpierw oblicza klucz strumieniowy:

$$z = 11001110000100111010,$$

dokładnie tak, jak w przykładzie 12.4. Dalej liczy różnice symetryczne bitów tego ciągu z kolejnymi bitami tekstu jawnego. Otrzymuje w ten sposób tekst zaszyfrowany:

$$y = 00011101010111010111,$$

który przekazuje Bolkowi. Oblicza również

$$s_{21} = s_{20}^2 \bmod n = 94739$$

i wysyła tę wartość Bolkowi.

Oczywiście Bolek zna rozkład  $n = 383 \cdot 503$ , więc także  $(p+1)/4 = 96$  oraz  $(q+1)/4 = 126$ . Rozpoczyna więc obliczenia:

$$\begin{aligned} a_1 &= ((p+1)/4)^{\ell+1} \bmod (p-1) \\ &= 96^{21} \bmod 382 \\ &= 266 \end{aligned}$$

oraz

$$\begin{aligned} a_2 &= ((q+1)/4)^{\ell+1} \bmod (q-1) \\ &= 126^{21} \bmod 502 \\ &= 486. \end{aligned}$$

Dalej,

$$\begin{aligned} b_1 &= s_{21}^{a_1} \bmod p \\ &= 94739^{266} \bmod 383 \\ &= 67 \end{aligned}$$

oraz

$$\begin{aligned} b_2 &= s_{21}^{a_2} \bmod q \\ &= 94739^{486} \bmod 503 \\ &= 126. \end{aligned}$$

Teraz Bolek przystępuje do rozwiązania układu kongruencji:

$$\begin{aligned} r &\equiv 67 \pmod{383} \\ r &\equiv 126 \pmod{503} \end{aligned}$$

i otrzymuje wynik, którym jest użyty przez Alicję załączek. Na tej podstawie odbudowuje jej klucz strumieniowy i na koniec oblicza różnice symetryczne bitów klucza strumieniowego z bitami tekstu zaszyfrowanego. W rezultacie otrzymuje na koniec pierwotny tekst jawnny.  $\square$

## 12.5. Uwagi i bibliografia

Obszerne omówienie generatorów bitów pseudolosowych można znaleźć w książce Kranakisa [KR86]; warto także poznać pracę przeglądową Lagariasa [LA90].

Pomysł generatora obcinającego pochodzi od Coppersmitha, Krawczyka i Mansoura [CKM94]. Gunther [GU88] podał inną praktyczną metodę budowania PRBG z użyciem LFSR. Metody łamania liniowego generatora kongruencyjnego opisał Boyar [BO89].

Podstawy teorii bezpiecznych generatorów bitów pseudolosowych opracował Yao [YA82], który wykazał uniwersalność testu następnego bitu. Dalsze podstawowe wyniki można znaleźć w pracy Bluma i Micaliego [BM84]. Generator BBS jest opisany w [BBS86]. Goldwasser i Micali [GM84] badali bezpieczeństwo problemu reszt kwadratowych; na ich pracy opiera się znaczna część materiału w punkcie 12.3.1. Przyjęliśmy tu jednak podejście Brassarda i Bratleya [BB88A, podrozdział 8.6], polegające na zmniejszaniu prawdopodobieństwa błędu neutralnego algorytmu typu Monte Carlo.

Właściwości generatora RSA były przedmiotem badań Alexiego, Chora, Goldreicha i Schnorra [ACGS88]. Generatorami bitów pseudolosowych opartymi na problemie logarytmu dyskretnego zajmowali się Blum i Micali [BM84], Long i Wiggerson [LW88] oraz Håstad, Schrift i Shamir [HSS93]. Vazirani i Vazirani [VV84] podali warunek wystarczający do bezpiecznego wybierania większej liczby bitów w każdej iteracji generatora bitów pseudolosowych.

Pomysł szyfrowania probabilistycznego pochodzi od Goldwassera i Micaliego [GM84]. Opis systemu kryptograficznego Bluma–Goldwassera znajduje się w [BG85].

---

## Ćwiczenia

- 12.1. Rozważamy liniowy generator kongruencyjny określony za pomocą warunku  $s_i = (as_{i-1} + b) \bmod M$ . Założmy, że  $M = qa + 1$ , gdzie  $a$  jest liczbą nieparzystą, a  $q$  parzystą, oraz niech  $b = 1$ . Wykaż, że algorytm przewidywania następnego bitu  $\mathbf{B}_i(z) = 1 - z$  dla  $i$ -tego bitu jest  $\epsilon$ -algorytmem, gdzie

$$\frac{1}{2} + \epsilon = \frac{q(a+1)}{2M}.$$

- 12.2. Oblicz pierwszych 100 bitów uzyskanych za pomocą generatora RSA, dla którego  $n = 36863$ ,  $b = 229$ , a założkiem jest  $s_0 = 25$ .
- 12.3. Na rysunku 12.11 jest opisany generator bitów pseudolosowych oparty na problemie logarytmu dyskretnego. Założmy, że  $p = 21383$ , elementem pierwotnym jest  $\alpha = 5$ , a założkiem jest  $s_0 = 15886$ . Oblicz pierwszych 100 bitów wyprodukowanych przez ten generator.
- 12.4. Założmy, że Bolek zna rozkład na czynniki pierwsze  $n = pq$  liczby  $n$  użytej w generatorze BBS.

Niech  $p$  będzie  $k$ -bitową liczbą pierwszą i niech  $\alpha$  będzie elementem pierwotnym modulo  $p$ .

Załążkiem  $x_0$  jest dowolny element  $\mathbb{Z}_p^*$ . Dla  $i \geq 0$  definiujemy

$$x_{i+1} = \alpha^{x_i} \pmod{p},$$

a następnie

$$f(x_0) = (z_1, z_2, \dots, z_\ell),$$

gdzie

$$z_i = \begin{cases} 1, & \text{gdy } x_i > p/2 \\ 0, & \text{gdy } x_i < p/2. \end{cases}$$

Wówczas  $f$  jest generatorem opartym na logarytmie dyskretnym o parametrach  $(k, \ell)$ .

### RYSUNEK 12.11. Generator oparty na logarytmie dyskretnym

- (a) Wyjaśnij, jak Bolek może wykorzystać tę wiedzę do obliczenia dowolnej wartości  $s_i$  z wartości  $s_0$  za pomocą  $2k$  mnożeń modulo  $\phi(n)$  oraz  $2k$  mnożeń modulo  $n$ , jeśli przyjąć, że liczba  $n$  ma  $k$  bitów w rozwinięciu binarnym. (Gdy  $i$  jest duże w porównaniu z  $k$ , takie podejście stanowi znaczące ulepszenie w stosunku do  $i$  mnożeń potrzebnych do kolejnego obliczenia  $s_0, \dots, s_i$ ).
  - (b) Użyj opisanej wyżej metody do obliczenia  $s_{10000}$ , jeśli  $n = 59701 = 227 \cdot 263$  oraz  $s_0 = 17995$ .
- 12.5. Udowodniliśmy, że do zmniejszenia prawdopodobieństwa błędu w neutralnym algorytmie Monte Carlo z  $1/2 - \epsilon$  do  $\delta$ , gdzie  $\delta + \epsilon < 1/2$ , wystarczy uruchomić algorytm  $m$  razy, gdzie

$$m = \left\lceil \frac{1 + \log_2 \delta}{\log_2(1 - 4\epsilon^2)} \right\rceil.$$

Wykaż, że wartość  $m$  jest rzędu  $O(1/(\delta\epsilon^2))$ .

- 12.6. Przypuśćmy, że Bolek otrzymał tekst zaszyfrowany w probabilistycznym systemie kryptograficznym Bluma–Goldwassera z kluczem publicznym. Oryginalny tekst jawny był napisany w języku angielskim. Każdy znak alfabetu został w naturalny sposób przekształcony w ciąg bitów o długości 5:  $A \leftrightarrow 00000$ ,  $B \leftrightarrow 00001, \dots, Z \leftrightarrow 11001$ . Tekst jawny składał się z 236 znaków alfabetu, powstał więc ciąg 1180 bitów. Następnie ten ciąg zaszyfrowano, a otrzymany tekst, dla oszczędności miejsca, zapisano w systemie heksadecymalnym. Ostateczny rezultat, tekst w zapisie heksadecymalnym, przedstawiamy w tablicy 12.4. Częścią tekstu zaszyfrowanego jest także wartość  $s_{1181} = 20291$ . Liczba  $n = 29893$  jest kluczem publicznym Bolka, a tajnym rozkładem tej liczby jest  $n = pq$ , gdzie  $p = 167$  i  $q = 179$ .

**TABLICA 12.4.** Tekst zaszyfrowany w systemie Bluma-Goldwassera

E1866663F17FDDBD1DC8C8FD2EEBC36AD7F53795DBA3C9CE22D  
C9A9C7E2A56455501399CA6B98AED22C346A529A09C1936C61  
ECDE10B43D226EC683A669929F2FFB912BFA96A8302188C083  
46119E4F61AD8D0829BD1CDE1E37DBA9BCE65F40C0BCE48A80  
0B3D087D76ECD1805C65D9DB730B8D0943266D942CF04D7D4D  
76BFA891FA21BE76F767F1D5DCC7E3F1D86E39A9348B3

Zadanie polega na odczytaniu danego tekstu zaszyfrowanego i odtworzeniu oryginalnego angielskiego tekstu jawnego, pochodzącego z „Under the Hammer” Johna Mortimera, Penguin Books, 1994.

# 13

---

## Dowody o wiedzy zerowej

---

### 13.1. Interakcyjne systemy dowodowe

Mówiąc nieformalnie, system dowodu o wiedzy zerowej pozwala przekonać interloktora o prawdziwości pewnego faktu bez ujawniania informacji o jego dowodzie. Na początek zajmijmy się interakcyjnym systemem dowodu. Uczestniczą w nim dwie osoby: Dorota i Stefan. Dorota jest osobą *dowodzącą*, Stefan jest *sprawdzającym*. Dorota zna pewne fakty i chce to udowodnić Stefanowi.

Musimy najpierw ustalić, jakie obliczenia wykonywane przez Dorotę i Stefana uznajemy za dopuszczalne, a także opisać przebieg interakcji. Wygodnie jest myśleć o obojgu jako o algorytmach probabilistycznych. Zarówno Dorota, jak i Stefan będą przeprowadzać prywatne obliczenia, a każde z nich dysponuje prywatnym generatorem liczb losowych. Porozumiewają się za pośrednictwem pewnego kanału komunikacyjnego. Na początku oboje mają daną wejściową  $x$ . Celem interakcyjnego dowodu, jaki stawia sobie Dorota, jest przekonanie Stefana o pewnej ustalonej własności  $x$ . Dokładniej,  $x$  będzie szczególnym przypadkiem określonego problemu decyzyjnego II, dla którego odpowiedź brzmi „tak” (czyli gdy  $x$  jest pozytywnym przypadkiem tego problemu).

Dowód interakcyjny, będący w istocie protokołem typu pytanie-odpowiedź, składa się z ustalonej liczby rund. Podczas każdej rundy Dorota i Stefan wykonują na zmianę następujące czynności:

- (1) odbierają wiadomość od partnera;
- (2) wykonują prywatne obliczenia;
- (3) wysyłają wiadomość do partnera.

Typowa runda protokołu składa się z pytania Stefana i odpowiedzi Doroty. Na końcu dowodu Stefan albo go *akceptuje*, albo *odrzuca*, w zależności od odpowiedzi udzielonych przez Dorotę. Protokół nazwiemy *interakcyjnym systemem dowodu* dla problemu decyzyjnego II, jeśli zawsze wtedy, gdy Stefan realizuje reguły protokołu, spełnione są następujące warunki:

**Zupełność.** Jeśli  $x$  jest pozytywnym przypadkiem problemu decyzyjnego  $\Pi$ , to Stefan akceptuje dowód Doroty.

**Adekwatność.** Jeśli  $x$  jest negatywnym przypadkiem problemu decyzyjnego  $\Pi$ , to prawdopodobieństwo akceptacji dowodu przez Stefana jest bardzo małe.

Ograniczymy się tu do omówienia interakcyjnych systemów dowodu, w których obliczenia Stefana mogą być wykonywane w czasie wielomianowym. Nie stawiamy natomiast żadnych ograniczeń na obliczenia Doroty (uznajemy ją za „wszechmocną”).

Zacznijmy od prezentacji interakcyjnego systemu dowodu dla **problemu nieizomorficzności grafów**. Problem **izomorficzności grafów** przedstawiamy na rysunku 13.1. Problem ten jest o tyle interesujący, że nie jest znany żaden algorytm o czasie wielomianowym, który by go rozwiązywał, choć z drugiej strony, nie ma dowodu na to, że jest on NP-zupełny.

**Dane** Dwa grafy o  $n$  wierzchołkach:  $G_1 = (V_1, E_1)$  i  $G_2 = (V_2, E_2)$ .

**Pytanie** Czy istnieje bijekcja  $\pi : V_1 \rightarrow V_2$ , taka że  $\{u, v\} \in E_1$  wtedy i tylko wtedy, gdy  $\{\pi(u), \pi(v)\} \in E_2$ ? (Innymi słowy, czy grafy  $G_1$  i  $G_2$  są izomorficzne?)

### RYSUNEK 13.1. Izomorficzność grafów

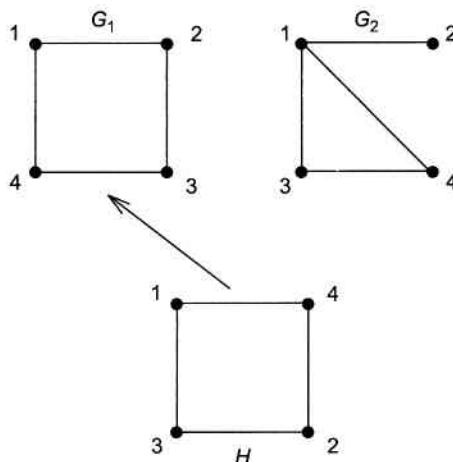
Omówimy interakcyjny system dowodu, który pozwoli Dorocie przekonać Stefana, że dane dwa grafy nie są izomorficzne. Dla uproszczenia przyjmijmy, że oba grafy  $G_1$  i  $G_2$  mają ten sam zbiór wierzchołków  $\{1, \dots, n\}$ .

Na rysunku 13.2 opisujemy interakcyjny system dowodu dla problemu nieizomorficzności grafów.

Dane wejściowe: dwa grafy  $G_1$  i  $G_2$ , każdy na zbiorze wierzchołków  $\{1, \dots, n\}$ .

- (1) powtórz następujące kroki  $n$  razy
- (2) Stefan wybiera losową liczbę całkowitą  $i = 1$  lub  $2$  oraz losową permutację  $\pi$  zbioru  $\{1, \dots, n\}$ ; oblicza  $H$ , obraz grafu  $G_i$  ze względu na permutację  $\pi$ , i wysyła graf  $H$  do Doroty
- (3) Dorota ustala wartość  $j$ , dla której  $H$  jest izomorficzny z  $G_j$ , i wysyła wartość  $j$  do Stefana
- (4) Stefan sprawdza, czy  $i = j$
- (5) Stefan akceptuje dowód Doroty, jeśli  $i = j$  w każdej z  $n$  rund

### RYSUNEK 13.2. Interakcyjny system dowodu nieizomorficzności grafów



RYSUNEK 13.3. Nieizomorficzne grafy Doroty i pytanie Stefana

A oto prosty przykład.

### Przykład 13.1

Niech  $G_1 = (V, E_1)$  i  $G_2 = (V, E_2)$ , gdzie  $V = \{1, 2, 3, 4\}$ ,  $E_1 = \{12, 14, 23, 34\}$  i  $E_2 = \{12, 13, 14, 34\}$ .

Załóżmy, że w kolejnej rundzie protokołu Stefan przekazuje Dorocie graf  $H = (V, E_3)$ , w którym  $E_3 = \{13, 14, 23, 24\}$  (patrz rysunek 13.3). Graf  $H$  jest izomorficzny z  $G_1$  (jednym z możliwych izomorfizmów jest permutacja  $(1\ 3\ 4\ 2)$ ). Dorota odpowiada „1”.  $\square$

Łatwo zauważyc, że ten przykładowy interakcyjny system dowodu spełnia warunki zupełności i adekwatności. Jeśli  $G_1$  nie jest izomorficzny z  $G_2$ , to w każdej rundzie będzie spełniona równość  $j = i$  i Stefan zaakceptuje dowód z prawdopodobieństwem 1. Tak więc protokół jest zupełny.

Z drugiej strony, założymy, że  $G_1$  jest izomorficzny z  $G_2$ . Wówczas dowolny graf  $H$  podany przez Stefana jako pytanie będzie izomorficzny z obydwojma grafami,  $G_1$  i  $G_2$ . Dorota nie ma podstaw, by ocenić, czy Stefan zbudował  $H$  jako izomorficzną kopię  $G_1$  czy  $G_2$ , nie pozostaje jej więc nic innego, jak w swojej odpowiedzi zgadywać:  $j = 1$  lub  $j = 2$ . Z kolei jedynym sposobem na to, by Stefan zaakceptował dowód jest odgadnięcie przez Dorotę wszystkich  $n$  wyborów wartości  $i$  dokonanych przez Stefana. Prawdopodobieństwo, że jej się to uda, jest równe  $2^{-n}$ . To potwierdza adekwatność algorytmu.

Warto podkreślić, że wszystkie obliczenia Stefana odbywają się w czasie wielomianowym. Nie możemy natomiast nic stwierdzić na temat czasu obliczeń Doroty, nie wiadomo bowiem, czy problem izomorficzności grafów jest rozwiążalny w czasie wielomianowym. Przyjęliśmy jednak, że Dorota dysponuje nieograniczoną mocą obliczeniową, zatem taka sytuacja mieści się w „regułach gry”.

### 13.2. Doskonałe dowody o wiedzy zerowej

Interakcyjne systemy dowodu są interesujące same w sobie, jednak najciekawszym rodzajem interakcyjnego dowodu jest dowód o wiedzy zerowej. To taki typ dowodu, przy którym Dorota przekonuje Stefana o pewnej własności  $x$ , ale po zakończeniu protokołu Stefan nadal nie potrafi sam udowodnić, że  $x$  tę własność posiada. Taka koncepcja niełatwo poddaje się formalnym definicjom, zacznijmy więc od przykładu.

Na rysunku 13.4 przedstawiamy interakcyjny dowód o wiedzy zerowej dla problemu izomorficzności grafów. Przykład pokaże, jak funkcjonuje odpowiedni protokół.

Dane wejściowe: dwa grafy  $G_1$  i  $G_2$ , każdy na zbiorze wierzchołków  $\{1, \dots, n\}$

- (1) powtóż następujące kroki  $n$  razy
- (2) Dorota wybiera losowo permutację  $\pi$  zbioru  $\{1, \dots, n\}$ ; oblicza  $H$ , obraz  $G_1$  ze względu na permutację  $\pi$ , i wysyła graf  $H$  do Stefana
- (3) Stefan wybiera losowo  $i = 1$  lub  $2$  i wysyła wybraną liczbę do Doroty
- (4) Dorota oblicza permutację  $\rho$  zbioru  $\{1, \dots, n\}$ , dla której  $H$  jest obrazem  $G_i$  ze względu na  $\rho$ , i wysyła  $\rho$  do Stefana  
(jeśli  $i = 1$ , Dorota definiuje  $\rho = \pi$ ; jeśli  $i = 2$ , Dorota określa  $\rho$  jako złożenie  $\sigma \circ \pi$ , gdzie  $\sigma$  jest pewną ustaloną permutacją przeprowadzającą  $G_2$  na  $G_1$ )
- (5) Stefan sprawdza, czy  $H$  jest obrazem  $G_i$  ze względu na  $\rho$
- (6) Stefan akceptuje dowód Doroty, jeśli  $H$  jest obrazem  $G_i$  w każdej z  $n$  rund

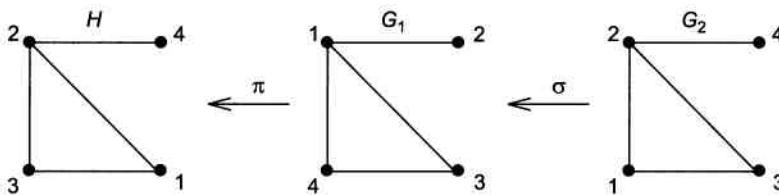
**RYSUNEK 13.4.** Interakcyjny doskonały system dowodu o wiedzy zerowej dla problemu izomorficzności grafów

#### Przykład 13.2

Niech  $G_1 = (V, E_1)$  i  $G_2 = (V, E_2)$ , gdzie  $V = \{1, 2, 3, 4\}$ ,  $E_1 = \{12, 13, 14, 34\}$  oraz  $E_2 = \{12, 13, 23, 24\}$ . Jednym z izomorfizmów z  $G_2$  na  $G_1$  jest permutacja  $\sigma = (4 \ 1 \ 3 \ 2)$ .

Przypuśćmy teraz, że w jednej z rund protokołu Dorota wybiera permutację  $\pi = (2 \ 4 \ 1 \ 3)$ . Wtedy zbiorem krawędzi grafu  $H$  jest zbiór  $\{12, 13, 23, 24\}$  (patrz rysunek 13.5).

Gdy Stefan formułuje pytanie  $i = 1$ , Dorota przekazuje mu permutację  $\pi$ , po czym Stefan sprawdza, czy obrazem grafu  $G_1$  wyznaczonym przez permutację  $\pi$  jest  $H$ . Jeśli pytaniem jest  $i = 2$ , Dorota przesyła Stefanowi złożenie  $\rho = \pi \circ \sigma = (3 \ 2 \ 1 \ 4)$  i Stefan sprawdza, czy  $H$  jest obrazem grafu  $G_2$  wyznaczonym przez  $\rho$ . □



RYSUNEK 13.5. Izomorficzne grafy Doroty

Łatwo sprawdzić zupełność i adekwatność protokołu. Widać, że gdy  $G_1$  jest izomorficzny z  $G_2$ , prawdopodobieństwo zaakceptowania dowodu przez Stefana jest równe 1. Jednakże jeśli  $G_1$  nie jest izomorficzny z  $G_2$ , jedynym sposobem Doroty na oszukanie Stefana jest poprawne odgadnięcie wartości i wybieranych przez Stefana w kolejnych rundach i wypisanie na taśmie komunikacyjnej (losowej) izomorficznej kopii grafu  $G_i$ . Prawdopodobieństwo odgadnięcia  $n$  losowych pytań Stefana jest równe  $2^{-n}$ .

Wszystkie obliczenia Stefana są wykonalne w czasie wielomianowym (będącym funkcją  $n$ , liczby wierzchołków w  $G_1$  i  $G_2$ ). Zauważmy też, choć nie jest to istotne, że również obliczenia Doroty mogą być wykonane w czasie wielomianowym, jeśli wie ona o istnieniu permutacji  $\sigma$ , która przeprowadza graf  $G_2$  na  $G_1$ .

Dlaczego nazywamy taki system dowodem o wiedzy zerowej? Otóż Stefan, choć przekonany o tym, że graf  $G_1$  jest izomorficzny z  $G_2$ , nie zyskuje żadnej wiedzy, która pozwoliłaby mu odkryć permutację  $\sigma$  przeprowadzającą  $G_2$  na  $G_1$ . W każdej rundzie dowodu widzi jedynie losową izomorficzną kopię  $H$  grafów  $G_1$  i  $G_2$  oraz permutację, która przekształca  $G_1$  na  $H$  albo  $G_2$  na  $H$  (ale nie obie naraz!). Ale do obliczenia losowych izomorficznych kopii tych grafów Stefan nie potrzebuje pomocy Doroty, może je obliczyć sam. Grafy  $H$  są wybierane niezależnie i losowo w każdej rundzie dowodu, więc wydaje się mało prawdopodobne, by mogło to ułatwić Stefanowi znalezienie izomorfizmu z  $G_1$  na  $G_2$ .

Przyjrzyjmy się dokładniej informacji, którą otrzymuje Stefan, uczestnicząc w interakcyjnym systemie dowodu. Pole widzenia Stefana w trakcie interakcyjnego dowodu możemy opisać w postaci *transkrypcji*, zawierającej następujące dane:

- (1) grafy  $G_1$  i  $G_2$ ;
- (2) wszystkie wiadomości przesyłane przez Dorotę i Stefana;
- (3) losowe liczby używane przez Stefana do generowania pytań.

Tak więc transkrypcja  $T$  wyżej przedstawionego interakcyjnego dowodu izomorficzności grafów miałaby następującą postać:

$$T = ((G_1, G_2); (H_1, i_1, \rho_1); \dots; (H_n, i_n, \rho_n)).$$

Sprawą zasadniczą z punktu widzenia formalnej definicji dowodu o wiedzy zerowej jest to, że Stefan (lub ktokolwiek inny) może falszować transkrypcje – nie

uczestnicząc w dowodzie interakcyjnym – tak aby „wyglądały” jak autentyczne. Warunek – grafy  $G_1$  i  $G_2$  muszą być izomorficzne. Algorytm fałszowania (ang. *forging algorithm*) transkrypcji znajduje się na rysunku 13.6. Jest to algorytm probabilistyczny działający w czasie wielomianowym. W żargonie teorii dowodów o wiedzy zerowej taki algorytm nazywa się *symulator*.

Dane wejściowe: dwa grafy  $G_1$  i  $G_2$ , każdy na zbiorze wierzchołków  $\{1, \dots, n\}$

- (1)  $T = (G_1, G_2)$
- (2) **for**  $j = 1$  **to**  $n$  **do**
- (3)     wybierz losowo  $i_j = 1$  lub  $2$
- (4)     wybierz losowo permutację  $\rho_j$  zbioru  $\{1, \dots, n\}$
- (5)     oblicz  $H_j$ , obraz  $G_{i_j}$  ze względu na  $\rho_j$
- (6)     dopisz  $(H_j, i_j, \rho_j)$  na koniec transkrypcji

**RYSUNEK 13.6.** Algorytm fałszowania transkrypcji dla problemu izomorficzności grafów

Fakt, że symulator potrafi fałszować transkrypcje, ma daleko sięgające konsekwencje. Wszystko, co Stefan (lub ktoś inny) może obliczyć na podstawie transkrypcji, można obliczyć także, używając transkrypcji fałszywej. Oznacza to, że udział Stefana w systemie dowodu nie zwiększa jego możliwości wykonywania obliczeń; w szczególności nie pozwala mu samodzielnie „udowodnić”, że grafy  $G_1$  i  $G_2$  są izomorficzne. Co więcej, pokazując transkrypcję  $T$ , Stefan nie jest w stanie przekonać kogokolwiek o izomorficzności grafów  $G_1$  i  $G_2$ , nie ma bowiem sposobu, by odróżnić transkrypcję autentyczną od sfałszowanej.

Powinniśmy jeszcze sprecyzować pogląd, że sfałszowana transkrypcja „wygląda” tak jak autentyczna. Ścisła definicja odwołuje się do pojęcia rozkładu prawdopodobieństwa.

### DEFINICJA 13.1

Załóżmy, że mamy interakcyjny system dowodu z czasem wielomianowym dla problemu decyzyjnego  $\Pi$  oraz symulator  $S$ , także z czasem wielomianowym. Niech  $T(x)$  oznacza zbiór wszystkich możliwych transkrypcji wynikłych w trakcie interakcyjnego dowodu przeprowadzanego przez Dorotę i Stefana na pozytywnym przypadku  $x$  problemu  $\Pi$  i niech  $\mathcal{F}(x)$  oznacza zbiór wszystkich możliwych fałszywych transkrypcji, które może wyprodukować  $S$ . Dla dowolnej transkrypcji  $T \in T(x)$  niech  $p_T(T)$  oznacza prawdopodobieństwo tego, że  $T$  jest transkrypcją powstałą w takcie interakcyjnego dowodu. Podobnie dla  $T \in \mathcal{F}(x)$  niech  $p_{\mathcal{F}}(T)$  będzie prawdopodobieństwem tego, że  $T$  jest (fałszywą) transkrypcją wyprodukowaną przez  $S$ . Przymijmy  $T(x) = \mathcal{F}(x)$  i założmy, że dla każdego  $T \in T(x)$  zachodzi  $p_T(T) = p_{\mathcal{F}}(T)$ . (Innymi słowy, zbiór autentycznych transkrypcji pokrywa się ze zbiorem transkrypcji fałszywych, a oba rozkłady prawdopodobieństwa są identyczne). Wówczas mówimy, że interakcyjny system dowodu jest *doskonałym systemem dowodu o wiedzy zerowej* dla Stefana.

Rzecz jasna możemy zdefiniować wiedzę zerową jak nam się żywnie spodoba. Chodzi jednak o to, żeby definicja uchwyciła nasze intuicyjne pojęcie o tym, czym owa „wiedza zerowa” być powinna. Twierdzimy, że interakcyjny system dowodu jest systemem o wiedzy zerowej dla Stefana, gdy istnieje symulator produkujący transkrypcje z rozkładem prawdopodobieństwa identycznym z tym, jaki pojawiłby się, gdyby to rzeczywiście Stefan brał udział w protokole. (Jest to pojęcie pokrewne, ale silniejsze od pojęcia nieodróżnialnego rozkładu prawdopodobieństwa, które omawialiśmy w rozdziale 12). Zauważaliśmy już, że transkrypcja zawiera pełną informację zdobytą przez Stefana w trakcie realizacji protokołu. Wydaje się zatem rozsądne założenie, że jeśli Stefan może coś zrobić dzięki uczestnictwu w protokole, to może on to zrobić równie dobrze, używając symulatora do wytworzenia fałszywej transkrypcji. Być może w ten sposób nie definiujemy „wiedzy”, ale czymkolwiek „wiedza” jest, Stefan nie zdobywa jej wcale!

Udowodnimy teraz, że interakcyjny system dowodu dla problemu izomorficzności grafów jest doskonałym systemem o wiedzy zerowej dla Stefana.

### TWIERDZENIE 13.1

Interakcyjny system dowodu dla problemu izomorficzności grafów jest systemem o wiedzy zerowej dla Stefana.

**DOWÓD** Niech  $G_1$  i  $G_2$  będą dwoma izomorficznymi grafami na  $n$ -elementowym zbiorze wierzchołków. Transkrypcja  $T$  (autentyczna lub fałszywa) zawiera  $n$  trójkę postaci  $(H, i, \rho)$ , gdzie  $i = 1$  lub  $2$ ,  $\rho$  jest permutacją zbioru  $\{1, \dots, n\}$ , a  $H$  jest obrazem  $G_1$  przy permutacji  $\rho$ . Taką trójkę nazwiemy *ważną*; niech  $\mathcal{R}$  oznacza zbiór wszystkich ważnych trójkę. Zaczniemy od obliczenia  $|\mathcal{R}|$ , liczby wszystkich ważnych trójkę. Oczywiście  $|\mathcal{R}| = 2 \cdot n!$ , gdyż każdy wybór  $i$  i  $\rho$  wyznacza jednoznacznie graf  $H$ .

W dowolnej rundzie  $j$  algorytmu fałszowania, jak łatwo spostrzec, każda ważna trójka występuje z jednakowym prawdopodobieństwem  $1/(2 \cdot n!)$ . Jakie jest prawdopodobieństwo tego, że ważna trójka  $(H, i, \rho)$  jest  $j$ -tą trójką w autentycznej transkrypcji? W interakcyjnym systemie dowodu Dorota pierwsza wybiera losową permutację  $\pi$ , a następnie oblicza  $H$  jako obraz grafu  $G_1$  przy permutacji  $\pi$ . Permutację  $\rho$  określa się jako  $\pi$ , gdy  $i = 1$ , oraz jako złożenie dwóch permutacji  $\pi$  i  $\sigma$ , gdy  $i = 2$ .

Zakładamy, że to Stefan wybiera losowo wartość  $i$ . Gdy  $i = 1$ , każda z  $n!$  permutacji  $\rho$  jest jednakowo prawdopodobna, ponieważ w tym przypadku  $\rho = \pi$ , a  $\pi$  jest permutacją wybraną losowo. Jednakże jeśli  $i = 2$ , to  $\rho = \pi \circ \sigma$ , gdzie  $\pi$  jest permutacją losową, a  $\sigma$  – ustaloną. Również w tym przypadku każda z możliwych permutacji  $\rho$  jest jednakowo prawdopodobna (niezależnie od wartości  $i$ ). Skoro  $i$  i  $\rho$  wyznaczają razem graf  $H$ , wszystkie trójkę w  $\mathcal{R}$  są także jednakowo prawdopodobne.

Z faktu, że transkrypcja składa się z konkatenacji  $n$  niezależnych losowych trójelek wynika, że

$$p_T(T) = p_{\mathcal{F}}(T) = \frac{1}{(2 \cdot n!)^n}$$

dla dowolnej transkrypcji  $T$ . ■

Dowód twierdzenia 13.1 zakłada, że Stefan, uczestnicząc w interakcyjnym systemie dowodu, wiernie realizuje protokół. Sytuacja staje się znacznie bardziej złożona, gdy tak nie jest. Czy jest tak, że interakcyjny dowód pozostaje dowodem o wiedzy zerowej nawet wtedy, gdy Stefan odchodzi od protokołu?

W przypadku problemu izomorficzności grafów Stefan może sprzeniewierzyć się protokołowi jedynie wtedy, gdy nie będzie wybierał wartości  $i$  losowo. Intuicyjnie, nie wydaje się, aby taki sposób postępowania przysparzał mu jakiekolwiek „wiedzy”. Jednakże wtedy transkrypcje wytworzone przez symulator nie będą „wyglądały” tak jak transkrypcje pochodzące od Stefana. Przypuśćmy, na przykład, że Stefan wybiera  $i = 1$  w każdej rundzie protokołu. Wówczas w transkrypcji interakcyjnego dowodu będziemy mieli  $i_j = 1$  dla  $1 \leq j \leq n$ , podczas gdy w transkrypcji pochodzącej z symulatora prawdopodobieństwo przyjęcia wartości  $i_j = 1$  dla  $1 \leq j \leq n$  jest równe  $2^{-n}$ .

Aby obejść tę trudność, musimy wykazać, że niezależnie od tego, jak bardzo „oszukujący” Stefan odchodzi od protokołu, istnieje symulator działający w czasie wielomianowym, który będzie produkował fałszywe transkrypcje „wygląające” jak transkrypcje utworzone przez Dorotę i (oszukującego) Stefana w trakcie interakcyjnego dowodu. Jak poprzednio, zwrot „wyglądać jak” uściślimy za pomocą pojęcia rozkładu prawdopodobieństwa.

Oto formalna definicja.

### DEFINICJA 13.2

Założmy, że mamy interakcyjny system dowodu z czasem wielomianowym dla problemu decyzyjnego  $\Pi$ . Niech  $V^*$  będzie dowolnym probabilistycznym algorytmem o czasie wielomianowym, używanym przez (być może oszukującego) sprawdzającego do generowania pytań. (Inaczej mówiąc,  $V^*$  reprezentuje albo uczciwego, albo oszukującego sprawdzającego). Oznaczmy przez  $T(V^*, x)$  zbiór wszystkich możliwych transkrypcji, które mogą wytworzyć Dorota i  $V^*$  w ramach interakcyjnego dowodu dla pozytywnego przypadku  $x$  problemu  $\Pi$ . Założymy dalej, że dla każdego  $V^*$  istnieje działający w oczekiwany czasie wielomianowym probabilistyczny algorytm  $S^* = S^*(V^*)$  (symulator) produkujący fałszywe transkrypcje. Niech  $\mathcal{F}(V^*, x)$  oznacza zbiór możliwych fałszywych transkrypcji. Dla każdej transkrypcji  $T \in T(V^*, x)$  niech  $p_{T, V^*}(T)$  oznacza prawdopodobieństwo tego, że  $T$  jest transkrypcją wytworzoną przez  $V^*$  w trakcie interakcyjnego dowodu. Podobnie, dla  $T \in \mathcal{F}(V^*, x)$  niech  $p_{\mathcal{F}, V^*}(T)$  będzie prawdopodobieństwem tego, że  $T$  jest fałszywą transkrypcją wytworzoną przez  $S^*$ . Przyjmijmy, że  $T(V^*, x) = \mathcal{F}(V^*, x)$  oraz dla każdego  $T \in T(V^*, x)$ ,  $p_{T, V^*}(T) = p_{\mathcal{F}, V^*}(T)$ . Wówczas interakcyjny system dowodu nazwiemy doskonałym systemem o wiedzy zerowej.

W szczególnym przypadku, gdy  $V^*$  to Stefan (czyli gdy Stefan działa uczciwie), powyższa definicja pokrywa się z definicją „doskonałego systemu o wiedzy zerowej dla Stefana”.

Do udowodnienia, że system dowodu jest doskonałym systemem o wiedzy zerowej, potrzebujemy ogólnego przekształcenia, które z dowolnego  $V^*$  zbuduje symulator  $S^*$ . Znajdziemy je dla systemu dowodu dla problemu izomorficzności grafów. Symulator będzie zastępował Dorotę, używając  $V^*$  jako „odnawialnego podprogramu”. Mówiąc nieformalnie,  $S^*$  stara się w każdej rundzie  $j$  odgadnąć pytanie  $i_j$ , które  $V^*$  zada w tej rundzie. Innymi słowy,  $S^*$  generuje losową ważną trójkę postaci  $(H_j, i_j, \rho_j)$ , po czym wykonuje algorytm  $V^*$ , by sprawdzić, jakie pytanie wynika z niego dla rundy  $j$ . Jeśli odgadnięta wartość  $i_j$  pokrywa się z pytaniem  $i'_j$  (wyprodukowanym przez  $V^*$ ), trójka  $(H_j, i_j, \rho_j)$  zostaje dołączona do fałszywej transkrypcji. Jeśli nie, trójka jest odrzucana,  $S^*$  odgaduje nowe pytanie  $i_j$ , a algorytm  $V^*$  zostaje ponownie uruchomiony po sprowadzeniu go do „stanu”, w którym znajdował się na początku bieżącej rundy. Termin „stan” oznacza tu wartości wszystkich zmiennych używanych przez algorytm.

Podamy teraz bardziej szczegółowy opis algorytmu symulacji  $S^*$ . W każdej chwili działania programu  $V^*$  bieżący stan  $V^*$  oznaczymy symbolem  $\text{stan}(V^*)$ . Opis algorytmu symulacji za pomocą pseudokodu przedstawiamy na rysunku 13.7.

Dane wejściowe: dwa grafy  $G_1$  i  $G_2$ , każdy na zbiorze wierzchołków  $\{1, \dots, n\}$

- (1)  $T = (G_1, G_2)$
- (2) **for**  $j = 1$  **to**  $n$  **do**
- (3)     zdefiniuj  $\text{starystan} = \text{stan}(V^*)$
- (4)     **repeat**
- (5)         wybierz losowo  $i_j = 1$  lub  $2$
- (6)         wybierz losową permutację  $\rho_j$  zbioru  $\{1, \dots, n\}$
- (7)         oblicz  $H_j$ , obraz  $G_i$  ze względu na  $\rho_j$
- (8)         wywołaj podprogram  $V^*$  z daną wejściową  $H_j$ ;
- (9)         wynik: pytanie  $i'_j$
- (10)        **if**  $i_j = i'_j$  **then**
- (11)            dołącz  $(H_j, i_j, \rho_j)$  na koniec  $T$
- (12)        **else**
- (13)            odnów  $V^*$ , definiując  $\text{stan}(V^*) = \text{starystan}$
- until**  $i_j = i'_j$

**RYSUNEK 13.7.** Algorytm fałszowania transkrypcji problemu izomorficzności grafów dla  $V^*$

Jeśli w żadnym momencie nie będzie spełniona równość  $i_j = i'_j$ , symulator może działać nieskończenie długo. Można jednak wykazać, że średni czas działania symulatora jest wielomianowy, a oba rozkłady prawdopodobieństwa  $p_{\mathcal{F}, V^*}(T)$  i  $p_{T, V^*}(T)$  są równe.

### TWIERDZENIE 13.2

Interakcyjny system dowodu dla problemu izomorficzności grafów jest doskonałym systemem o wiedzy zerowej.

**DOWÓD** Zauważmy, po pierwsze, że niezależnie od tego, jak  $V^*$  generuje pytania, prawdopodobieństwo tego, że wybór  $i_j$  dokonany przez  $S^*$  pokrywa się z pytaniem  $i'_j$  jest równe  $1/2$ . Tak więc średnio licząc,  $S^*$  wyprodukuje dwie trójkę na każdą trójkę dołączoną do fałszywej transkrypcji. Dlatego średni czas działania symulatora jest wielomianowy ze względu na  $n$ .

Trudniej będzie wykazać, że oba rozkłady prawdopodobieństwa  $p_{\mathcal{F},V^*}(T)$  i  $p_{T,V^*}(T)$  są równe. W twierdzeniu 13.1 mogliśmy obliczyć oba te rozkłady i zobaczyć, że są jednakowe – przy założeniu uczciwości Stefana. Odwoaliśmy się wówczas także do tego, że trójkę  $(H, i, \rho)$  generowane w kolejnych rundach dowodu są niezależne. Jednakże teraz, w innych warunkach, nie potrafimy obliczyć wprost rozkładów prawdopodobieństwa. Co więcej, trójkę powstającą w kolejnych rundach dowodu nie muszą już być niezależne. Na przykład, pytanie przedstawione przez  $V^*$  w rundzie  $j$  może zależeć – i to w sposób bardzo złożony – od pytań z poprzednich rund i od odpowiedzi Doroty.

Aby poradzić sobie z tymi trudnościami, przyjrzymy się rozkładom prawdopodobieństwa na możliwych częściowych transkrypcjach, powstających w trakcie symulacji lub interakcyjnego dowodu, i na tej podstawie przeprowadźmy dowód indukcyjny ze względu na liczbę rund. Dla  $0 \leq j \leq n$  określamy rozkłady prawdopodobieństwa  $p_{T,V^*,j}$  oraz  $p_{\mathcal{F},V^*,j}$  na zbiorze częściowych transkrypcji  $\mathcal{T}_j$ , które mogą pojawić się na końcu rundy  $j$ . Zauważmy, że  $p_{T,V^*,n} = p_{T,V^*}$  i  $p_{\mathcal{F},V^*,n} = p_{\mathcal{F},V^*}$ . Jeśli zatem potrafimy wykazać, że oba rozkłady  $p_{T,V^*,j}$  i  $p_{\mathcal{F},V^*,j}$  pokrywają się dla każdego  $j$ , będziemy mogli wywnioskować stąd tezę.

Przypadek  $j = 0$  odpowiada początkowi algorytmu. Na tym etapie transkrypcja zawiera tylko dwa grafy,  $G_1$  i  $G_2$ , a więc rozkłady prawdopodobieństwa są równe. Niech ten przypadek stanowi pierwszy krok indukcyjny.

Przymijmy następujące założenie indukcyjne: dla pewnego  $j \geq 1$  rozkłady prawdopodobieństwa  $p_{T,V^*,j-1}$  i  $p_{\mathcal{F},V^*,j-1}$  na  $\mathcal{T}_j$  są równe. Wykażemy, że równe są także rozkłady  $p_{T,V^*,j}$  i  $p_{\mathcal{F},V^*,j}$  na  $\mathcal{T}_j$ .

Zastanówmy się, co się dzieje podczas rundy  $j$  interakcyjnego dowodu. Prawdopodobieństwo tego, że  $i'_j = 1$  (gdzie  $i'_j$  jest pytaniem sformułowanym przez  $V^*$ ) jest pewną liczbą rzeczywistą  $p_1$ , a wtedy prawdopodobieństwo tego, że  $i'_j = 2$  jest równe  $1 - p_1$ . Wartość  $p_1$  zależy od stanu algorytmu  $V^*$  na początku rundy  $j$ . Stwierdziliśmy uprzednio, że w interakcyjnym dowodzie Dorota wybiera każdy możliwy graf  $H$  z jednakowym prawdopodobieństwem. Podobnie każda permutacja  $\rho$  występuje z jednakowym prawdopodobieństwem, niezależnym od wartości  $p_1$ , gdyż dla każdego z możliwych wyborów  $i'_j$  wszystkie permutacje są jednakowo prawdopodobne. W rezultacie prawdopodobieństwo tego, że  $j$ -tą trójką w transkrypcji jest  $(H, i, \rho)$ , równa się  $p_1/n!$  dla  $i = 1$  oraz  $(1 - p_1)/n!$  dla  $i = 2$ .

Dokonajmy podobnej analizy symulacji. W każdej iteracji pętli **repeat**  $S^*$  wybiera graf  $H$  z prawdopodobieństwem  $1/n!$ . Prawdopodobieństwo tego, że

$i_j = 1$  i jednocześnie 1 jest pytaniem zadanym przez  $V^*$ , jest równe  $p_1/2$ , a prawdopodobieństwo tego, że  $i_j = 2$  i  $V^*$  zadaje pytanie 2, jest równe  $(1 - p_1)/2$ . W każdej z tych sytuacji  $(H, i, \rho)$  występuje w transkrypcji jako  $j$ -ta trójka. Z prawdopodobieństwem  $1/2$  w dowolnej danej iteracji pętli **repeat** nic nie zostanie zapisane na taśmie.

Rozważmy najpierw przypadek  $i_j = 1$ . Jak stwierdziliśmy wyżej, prawdopodobieństwo tego, że 1 jest również wyborem  $V^*$ , jest równe  $p_1$ . Prawdopodobieństwo zapisania w transkrypcji  $(H, i, \rho)$  jako  $j$ -tej trójki w trakcie  $\ell$ -tej iteracji pętli **repeat** jest równe

$$\frac{p_1}{2^\ell \cdot n!}.$$

Tak więc prawdopodobieństwo tego, że  $(H, i, \rho)$  jest  $j$ -tą trójką w transkrypcji, jest równe

$$\frac{p_1}{2 \cdot n!} \left( 1 + \frac{1}{2} + \frac{1}{4} + \dots \right) = \frac{p_1}{n!}.$$

Dla przypadku  $i_j = 2$  rozumujemy analogicznie: prawdopodobieństwo wpisania  $(H, i, \rho)$  do transkrypcji jako  $j$ -tej trójki jest równe  $(-p_1)/n!$ .

Okazuje się więc, że oba rozkłady prawdopodobieństwa na częściowych transkrypcjach na końcu rundy  $j$  są takie same. Stosując indukcję, wniosimy, że oba rozkłady prawdopodobieństwa, czyli  $p_{\mathcal{F}, V^*}(T)$  i  $p_{\mathcal{T}, V^*}(T)$ , są identyczne, co kończy dowód. ■

Warto przyjrzeć się także interakcyjnemu systemowi dowodu dla problemu nieizomorficzności grafów. Nietrudno wykazać, że dowód jest dowodem o wiedzy zerowej, jeśli Stefan realizuje wiernie protokół (czyli wybiera losowo każdą stanowiącą pytanie izomorficzną kopię grafu  $G_i$ , przy czym wartości  $i = 1$  i  $i = 2$  są także wybierane losowo). Dalej, przyjmując, że Stefan tworzy swoje grafy-pytania jako izomorficzne kopie grafów  $G_1$  lub  $G_2$ , protokół pozostaje protokołem o wiedzy zerowej nawet wtedy, gdy nie wybiera on pytań losowo. Przypuśćmy jednak, że nasz wszechobecny przeciwnik Oskar podsuwa Stefanowi graf  $H$  izomorficzny z jednym z grafów  $G_1$  i  $G_2$ , ale ten nie wie, z którym. Jeśli teraz Stefan użyje tego grafu  $H$  w charakterze pytania w interakcyjnym systemie dowodu, Dorota odpowie mu izomorfizmem, którego wcześniej nie znał i przypuszczał nigdy by sam nie wymyślił. W tej sytuacji system dowodu przestaje być (intuicyjnie) systemem o wiedzy zerowej i nie wydaje się prawdopodobne, by symulator był w stanie sfałszować transkrypcję.

Można tak zmodyfikować dowód izomorficzności grafów, by stał się doskonały dowodem o wiedzy zerowej, ale nie zamierzamy tu wchodzić w szczegóły.

Przytoczymy teraz kilka innych przykładów doskonałych dowodów o wiedzy zerowej. Na rysunku 13.8 przedstawiamy doskonały dowód o wiedzy zerowej dla reszt kwadratowych (modulo  $n = pq$ , gdzie  $p$  i  $q$  są liczbami pierwszymi).

Celem Doroty jest udowodnienie, że  $x$  jest resztą kwadratową. W każdej rundzie generuje ona losową resztę kwadratową  $y$  i przesyła ją Stefanowi. Następnie, zależnie od jego pytania, odpowiada mu pierwiastkiem kwadratowym albo z  $y$ , albo z  $xy$ .

Dane wejściowe: liczba całkowita  $n$  z nieznanym rozkładem  
 $n = pq$ , gdzie  $p$  i  $q$  są liczbami pierwszymi, oraz  $x \in \text{QR}(n)$

(1) powtóż następne kroki  $\log_2 n$  razy:

(2) Dorota wybiera losowo  $v \in \mathbb{Z}_n^*$  i oblicza

$$y = v^2 \pmod{n}$$

po czym wysyła  $y$  do Stefana

(3) Stefan wybiera losowo liczbę całkowitą  $i = 0$  lub  $1$  i przesyła ją Dorocie

(4) Dorota oblicza

$$z = u^i v \pmod{n},$$

gdzie  $u$  jest pierwiastkiem kwadratowym z  $x$ , po czym przesyła Stefanowi wartość  $z$

(5) Stefan sprawdza, czy

$$z^2 \equiv x^i y \pmod{n}$$

(6) Stefan akceptuje dowód Doroty, jeśli w każdej z  $\log_2 n$  rund obliczenie w kroku 5 daje wynik pozytywny

**RYSUNEK 13.8.** Interakcyjny doskonały system dowodu o wiedzy zerowej dla reszt kwadratowych

Taki protokół jest oczywiście zupełny. Przystępując do dowodu jego adekwatności, zauważmy, że jeśli  $x$  nie jest resztą kwadratową, to Dorota może odpowiedzieć tylko na jedno z dwóch możliwych pytań, gdyż  $y$  jest resztą kwadratową wtedy i tylko wtedy, gdy  $xy$  nią nie jest. Tak więc Dorota zostanie przyłapana na oszustwie z prawdopodobieństwem  $1/2$ , podczas gdy prawdopodobieństwo oszukania Stefana we wszystkich  $\log_2 n$  rundach jest równe tylko  $2^{-\log_2 n} = 1/n$ . (Liczba  $\log_2 n$  rund wynika z tego, że rozmiar problemu jest proporcjonalny do liczby bitów w binarnej reprezentacji liczby  $n$ , a tych jest właśnie  $\log_2 n$ ). Wiadomo, że prawdopodobieństwo skutecznego oszustwa maleje wykładniczo wraz ze wzrostem rozmiaru problemu, podobnie jak w dowodzie izomorficzności grafów o wiedzy zerowej).

O tym, że system jest doskonałym systemem dowodu o wiedzy zerowej dla Stefana, możemy się przekonać podobnie jak w przypadku problemu izomorficzności grafów. Stefan może wygenerować trójkę  $(y, i, z)$ , wybierając najpierw  $i$  oraz  $z$ , a następnie definiując

$$y = z^2(x^i)^{-1} \pmod{n}.$$

Wygenerowane w ten sposób trójki mają dokładnie taki sam rozkład prawdopodobieństwa jak trójki powstające w ramach protokołu, jeśli przyjmiemy, że Stefan generuje pytania losowo. Wykazanie, że system jest doskonałym systemem o wiedzy zerowej (dla dowolnego  $V^*$ ) przebiega podobnie jak dla problemu izomorficzności grafów. Wymaga ono zbudowania symulatora  $S^*$  odgadującego pytania formułowane przez  $V^*$  i zachowującego tylko trójki odpowiadające prawidłowo odgadniętym pytaniom.

Przedstawimy teraz jeszcze jeden przykład doskonałego dowodu o wiedzy zerowej, tym razem dla problemu decyzyjnego związanego z problemem logarytmu dyskretnego. Problem, który nazwiemy **problem przynależności do podgrupy**, formułujemy na rysunku 13.9. Oczywiście, jeśli taka liczba całkowita  $k$  istnieje, jest ona po prostu logarytmem dyskretnym  $\beta$ .

**Dane** Dwie dodatnie liczby całkowite  $n$  i  $\ell$  oraz dwa różne elementy  $\alpha, \beta \in \mathbb{Z}_n^*$ , gdzie  $\alpha$  ma w  $\mathbb{Z}_n^*$  rząd  $\ell$ .

**Pytanie** Czy istnieje liczba całkowita  $k$ , taka że  $\beta = \alpha^k$ ? (Innymi słowy, czy  $\beta$  jest elementem podgrupy grupy  $\mathbb{Z}_n^*$  generowanej przez  $\alpha$ ?)

RYSUNEK 13.9. Problem przynależności do podgrupy

Dane wejściowe: dodatnia liczba całkowita  $n$  oraz dwa różne elementy  $\alpha, \beta \in \mathbb{Z}_n^*$ , gdzie  $\alpha$  ma w  $\mathbb{Z}_n^*$  publicznie znany rząd  $\ell$

- (1) powtórz następujące kroki  $\log_2 n$  razy:
  - (2) Dorota wybiera losowo  $j$ ,  $0 \leq j \leq \ell - 1$ , oblicza
- $$\gamma = \alpha^j \text{ mod } n,$$
- (3) po czym wysyła  $y$  do Stefana
  - (4) Stefan wybiera losowo liczbę całkowitą  $i = 0$  lub  $1$  i wysyła ją do Doroty
  - (5) Dorota oblicza
- $$h = j + ik \text{ mod } \ell,$$
- gdzie
- $$k = \log_\alpha \beta,$$
- (6) i wysyła  $h$  do Stefana
  - (5) Stefan sprawdza, czy
- $$\alpha^h \equiv \beta^i \gamma \pmod{n}$$
- (6) Stefan akceptuje dowód Doroty, jeśli obliczenie w kroku 5 daje pozytywny wynik w każdej z  $\log_2 n$  rund

RYSUNEK 13.10. Doskonały interakcyjny system dowodu o wiedzy zerowej dla problemu przynależności do podgrupy

Na rysunku 13.10 przedstawiamy doskonały dowód o wiedzy zerowej dla tego problemu. Analiza tego protokołu przebiega podobnie jak w poprzednich omówionych tu przypadkach. Szczegóły pozostawiamy Czytelnikowi.

---

### 13.3. Schemat wiążącego bitu

System dowodu izomorficzności grafów o wiedzy zerowej jest bardzo interesujący, ale przydatniejsze byłyby dowody o wiedzy zerowej dla problemów uznanych za NP-zupełne. Teoria dostarcza argumentów wskazujących na to, że nie istnieją doskonałe dowody o wiedzy zerowej dla problemów NP-zupełnych, możemy jednak opisać systemy dowodu o nieco słabszej własności, zwanej *obliczeniową wiedzą zerową*. Istniejące systemy dowodu tego typu omówimy w następnym podrozdziale, tu zajmiemy się metodą wiążącego bitu, kluczowym narzędziem w systemie dowodu.

Przypuśćmy, że Dorota zapisuje wiadomość na kartce papieru, po czym umieszcza ją w sejfie (którego kombinację zna) i przekazuje sejf Stefanowi. Choć Stefan nie pozna treści wiadomości przed otwarciem sejfu, zgodzimy się, że Dorota pozostaje nią *związana*, ponieważ nie może jej zmienić. Z kolei Stefan, jeśli nie zna odpowiedniej kombinacji, nie jest w stanie odczytać wiadomości, o ile Dorota nie otworzy mu sejfu. (Przypomnijmy, że w rozdziale 4 użyliśmy podobnej analogii do przedstawienia systemu kryptograficznego z kluczem publicznym, jednak wtedy osobą, która mogła otworzyć sejf był odbiorca wiadomości, Stefan).

Załóżmy, że treścią wiadomości jest jeden bit  $b = 0$  lub  $b = 1$ , w jakiś sposób zaszyfrowany przez Dorotę. Tę zaszyfrowaną postać bitu nazywa się często *klekssem* (ang. *blob*)<sup>†</sup>, natomiast metodę szyfrowania określa się mianem **schematu wiążącego bitu** (ang. *bit commitment scheme*). Ogólniej, schemat wiążącego bitu można określić jako funkcję  $f : \{0, 1\} \times X \rightarrow Y$ , gdzie  $X$  i  $Y$  są zbiorami skończonymi. Szyfrem  $b$  jest dowolna wartość  $f(b, x)$  dla  $x \in X$ . Schemat wiążącego bitu powinien być (mówiąc nieformalnie):

**Tajny.** Niezależnie od tego, czy  $b = 0$  czy  $b = 1$ , Stefan nie może odtworzyć wartości  $b$  z kleksa  $f(b, x)$ .

**Zobowiązujący.** Dorota może w późniejszym czasie „usunąć” kleks, ujawniając wartość  $x$  użytą podczas szyfrowania, aby przekonać Stefana, iż to właśnie  $b$  było zaszyfrowanym bitem. Nie powinna ona jednak mieć możliwości uzyskania z kleksa obu wartości 0 i 1.

Jeśli Dorota zechce zobowiązać się co do ciągu bitów, po prostu zobowiąże się co do każdego bitu z osobna.

---

<sup>†</sup>Spotyka się również określenie „obiekt bitowy” (przyp. red.).

Do realizacji schematu wiążącego bitu może posłużyć probabilistyczny system kryptograficzny Goldwassera–Micaliego opisany w podrozdziale 12.4. Przypomnijmy, że mamy w nim liczbę  $n = pq$  dla pewnych liczb pierwszych  $p$  i  $q$  oraz liczbę  $m \in \widetilde{\text{QR}}(n)$ . Liczby całkowite  $n$  i  $m$  są publicznie znane, podczas gdy rozkład  $n = pq$  zna tylko Dorota. W naszym schemacie wiążącego bitu mamy  $X = Y = \mathbb{Z}_n^*$  oraz

$$f(b, x) = m^b x^2 \pmod{n}.$$

Szyfrując wartość  $b$ , Dorota wybiera losowo  $x$  i oblicza  $y = f(b, x)$ ; kleksem jest wtedy wartość  $y$ .

Gdy później Dorota zdecyduje się usunąć  $y$ , ujawni wartości  $b$  i  $x$ . Stefan sprawdzi wówczas, że

$$y \equiv m^b x^2 \pmod{n}.$$

Zatrzymajmy się teraz nad właściwościami tajności i zobowiązania. Kleks ukrywa albo 0, albo 1, nie ujawniając żadnej informacji o wartości tekstu jawnego  $x$ , pod warunkiem że problem reszt kwadratowych jest obliczalnie nieroziągalny (szeroko to omawialiśmy w rozdziale 12). Schemat jest więc tajny.

Czy jest zobowiązujący? Przypuśćmy, że nie. Wtedy

$$mx_1^2 \equiv x_2^2 \pmod{n}$$

dla pewnych  $x_1, x_2 \in \mathbb{Z}_n^*$ . Ale wówczas

$$m \equiv (x_2 x_1^{-1})^2 \pmod{n},$$

co jest sprzeczne z założeniem  $m \in \widetilde{\text{QR}}(n)$ .

Schematy wiążącego bitu okażą się przydatne w budowie dowodów o wiedzy zerowej. Mają one jednak także inne ciekawe zastosowanie, a mianowicie do rzutów monetą przez telefon. Założymy, że Alicja i Bolek chcą podjąć decyzję na podstawie losowego rzutu monetą, ale znajdują się daleko od siebie, co oznacza, że nie jest możliwe, by jedno z nich fizycznie rzuciło monetą, a drugie sprawdziło wynik. Wyjściem z tego dylematu jest schemat wiążącego bitu. Jedno z dwóch, na przykład Alicja, wybiera losowo bit  $b$ , oblicza dla niego kleks  $y$  i przekazuje go drugiemu, a więc Bolekowi. Teraz Bolek odgaduje wartość  $b$ , a wtedy Alicja usuwa kleks i ujawnia  $b$ . Tajność schematu zapewnia, że Bolek nie jest w stanie obliczyć wartości  $b$  na podstawie kleksa  $y$ , natomiast fakt, że jest on zobowiązujący, oznacza, że Alicja nie może zmienić zdania po tym, jak Bolek ujawni swoje domniemanie.

A oto inny przykład schematu wiążącego bitu, tym razem oparty na problemie logarytmu dyskretnego. Przypomnijmy z punktu 5.1.2, że jeśli  $p \equiv 3 \pmod{4}$  jest liczbą pierwszą, taką że problem logarytmu dyskretnego jest obliczeniowo nieroziągalny w  $\mathbb{Z}_p^*$ , to drugi najmniej znaczący bit logarytmu dyskretnego jest bezpieczny. W rzeczywistości udowodniono dla liczb pierwszych  $p \equiv 3 \pmod{4}$ ,

że do rozwiązywania problemu logarytmu dyskretnego w  $\mathbb{Z}_p^*$  można użyć dowolnego algorytmu typu Monte Carlo dla problemu drugiego bitu z prawdopodobieństwem błędu równym  $1/2 - \epsilon$  dla  $\epsilon > 0$ . Ten znacznie mocniejszy wynik daje podstawę do zastosowania schematu bitu wiążącego.

W tym schemacie bitu wiążącego  $X = \{1, \dots, p-1\}$  oraz  $Y = \mathbb{Z}_p^*$ . Drugi najmniej znaczący bit liczby całkowitej  $x$ , który oznaczymy symbolem  $\text{SLB}(x)$ , definiujemy w sposób następujący:

$$\text{SLB}(x) = \begin{cases} 0, & \text{gdy } x \equiv 0, 1 \pmod{4}, \\ 1, & \text{gdy } x \equiv 2, 3 \pmod{4}. \end{cases}$$

Schemat wiążącego bitu  $f$  określmy teraz tak:

$$f(b, x) = \begin{cases} \alpha^x \pmod{p}, & \text{gdy } \text{SLB}(x) = b, \\ \alpha^{p-x} \pmod{p}, & \text{gdy } \text{SLB}(x) \neq b. \end{cases}$$

Innymi słowy, wybieramy losowo liczbę, której drugim najmniej znaczącym bitem jest  $b$ , podnosimy  $\alpha$  do tej potęgi modulo  $p$  – i to jest szyfrem bitu  $b$ . Zauważmy, że  $\text{SLB}(p-x) \neq \text{SLB}(x)$ , ponieważ  $p \equiv 3 \pmod{4}$ .

Schemat jest zobowiązujący, a jak wynika z powyższych uwag, także tajny, pod warunkiem, że problem logarytmu dyskretnego jest obliczeniowo nierozerwialny w  $\mathbb{Z}_p^*$ .

### 13.4. Dowody o obliczeniowej wiedzy zerowej

Opiszemy tu system dowodu o wiedzy zerowej dla NP-zupełnego problemu **3-kolorowalności grafu** przedstawionego na rysunku 13.11. W systemie dowodu korzysta się ze schematu wiążącego bitu. Dokładniej, zastosujemy schemat wiążącego bitu opisany w podrozdziale 13.3, oparty na szyfrowaniu probabilistycznym. Zakładamy, że Dorota zna 3-kolorowanie  $\phi$  grafu  $G$  i chce za pomocą dowodu o wiedzy zerowej przekonać Stefana, że  $G$  jest 3-koloralny. Bez utraty ogólności możemy założyć, że wierzchołki grafu  $G$  stanowią zbiór  $V = \{1, \dots, n\}$ . Niech  $m = |E|$ . Do opisu systemu dowodu użyjemy schematu wiążącego bitu  $f : \{0, 1\} \times X \rightarrow Y$ , który jest znany publicznie. Tym razem chcemy szyfrować

**Dane** Graf  $G = (V, E)$  o  $n$  wierzchołkach.

**Pytanie** Czy istnieje właściwe 3-kolorowanie grafu  $G$ ? (W języku matematyki, czy istnieje funkcja  $\phi : V(G) \rightarrow \{1, 2, 3\}$ , taka że jeśli  $\{u, v\} \in E$ , to  $\phi(u) \neq \phi(v)$ ?)

RYSUNEK 13.11. 3-kolorowalność grafu

nie bity, lecz kolory, zatem zastąpimy kolor 1 dwoma bitami 01, kolor 2 zapiszemy jako 10, kolor 3 zaś jako 11. Każdy z dwóch bitów składających się na kod koloru zaszyfrujemy funkcją  $f$ .

Interakcyjny system dowodu przedstawiamy na rysunku 13.12. Nieformalnie mówiąc, jest on realizowany w następujący sposób. Dorota zobowiązuje się co do kolorowania, które jest permutacją ustalonego kolorowania  $\phi$ . Stefan żąda od

Dane wejściowe: graf  $G = (V, E)$  na zbiorze wierzchołków  $\{1, \dots, n\}$

- (1) powtórz następujące kroki  $m^2$  razy:
- (2) niech  $\phi$  będzie 3-kolorowaniem grafu  $G$ ; Dorota wybiera losowo permutację  $\pi$  zbioru  $\{1, 2, 3\}$ ; dla  $1 \leq i \leq n$  definiuje

$$c_i = \pi(\phi(i))$$

i zapisuje  $c_i$  jako ciąg bitowy o długości dwa

$$c_i = c_{i,1}c_{i,2}$$

następnie dla każdego  $1 \leq i \leq n$  wybiera losowo dwa elementy  $r_{i,1}, r_{i,2} \in X$  i oblicza

$$R_{i,j} = f(c_{i,j}, r_{i,j}),$$

$j = 1, 2$ ; wysyła Stefanowi listę

$$(R_{1,1}, R_{1,2}, \dots, R_{n,1}, R_{n,2})$$

- (3) Stefan wybiera losowo krawędź  $\{u, v\} \in E$  i wysyła ją do Doroty

- (4) Dorota wysyła Stefanowi  $(c_{u,1}, c_{u,2}, r_{u,1}, r_{u,2})$  i  $(c_{v,1}, c_{v,2}, r_{v,1}, r_{v,2})$

- (5) Stefan sprawdza, czy

$$(c_{u,1}, c_{u,2}) \neq (c_{v,1}, c_{v,2})$$

$$(c_{u,1}, c_{u,2}) \neq (0, 0)$$

$$(c_{v,1}, c_{v,2}) \neq (0, 0)$$

$$R_{u,j} = f(c_{u,j}, r_{u,j}), \quad j = 1, 2$$

$$R_{v,j} = f(c_{v,j}, r_{v,j}), \quad j = 1, 2$$

- (6) Stefan akceptuje dowód Doroty, jeśli obliczenie w kroku 5 daje pozytywny wynik w każdej z  $m^2$  rund

**RYSUNEK 13.12.** Interakcyjny system dowodu 3-kolorowalności grafu o obliczeniowej wiedzy zerowej

niej, by usunęła kleks odpowiadający wierzchołkom końcowym pewnej losowo wybranej krawędzi. Dorota to robi, po czym Stefan sprawdza, czy bity wiążące są takie, jak twierdzi Dorota, i czy kolory są różne. Podkreślimy, że wszystkie obliczenia Stefana odbywają się w czasie wielomianowym, podobnie jak obliczenia Doroty, jeśli tylko zna ona choćby jedno 3-kolorowanie  $\phi$ .

Popatrzmy na bardzo prosty przykład.

*Przykład 13.3*

Niech  $G = (V, E)$ , gdzie

$$V = \{1, 2, 3, 4, 5\}$$

oraz

$$E = \{12, 14, 15, 23, 34, 45\}.$$

Załóżmy, że Dorota zna 3-kolorowanie  $\phi$ , takie że  $\phi(1) = 1$ ,  $\phi(2) = \phi(4) = 2$  i  $\phi(3) = \phi(5) = 3$ . Przyjmijmy także, iż schemat wiążącego bitu jest określony wzorem  $f(b, x) = 156897^b x^2 \pmod{321389}$ , gdzie  $b = 0, 1$  oraz  $x \in \mathbb{Z}_{321389}^*$ .

Przypuśćmy, że w pewnej rundzie dowodu Dorota wybiera permutację  $\pi = (1\ 3\ 2)$ . Wówczas oblicza:

$$c_1 = 1$$

$$c_2 = 3$$

$$c_3 = 2$$

$$c_4 = 3$$

$$c_5 = 2.$$

Otrzymane kolorowanie zostanie zakodowane w systemie binarnym w postaci 10-wyrazowego ciągu bitów

$$0111101110,$$

dla których zostaną obliczone odpowiednie kleksy. Niech wygląda to tak, jak w poniższej tablicy:

| $b$ | $x$    | $f(b, x)$ |
|-----|--------|-----------|
| 0   | 147658 | 176593    |
| 1   | 318856 | 205585    |
| 1   | 14497  | 189102    |
| 1   | 285764 | 294039    |
| 1   | 128589 | 230968    |
| 0   | 228569 | 77477     |
| 1   | 53369  | 305090    |
| 1   | 194634 | 276484    |
| 1   | 202445 | 292707    |
| 0   | 177561 | 290599    |

Teraz Dorota przekazuje Stefanowi dziesięć obliczonych w ten sposób wartości  $f(b, x)$ .

Założymy dalej, że Stefan pyta o krawędź 34. Wtedy Dorota usuwa cztery kleksy: dwa odpowiadające wierzchołkowi 3 i dwa odpowiadające wierzchołkowi 4. Ujawnia zatem Stefanowi cztery pary uporządkowane:

$$(b, x) = (1, 128589), (0, 228569), (1, 53369), (1, 194634).$$

Stefan najpierw sprawdza, czy kolory są różne. Ciąg 10 koduje kolor 2, ciąg 11 jest kodem koloru 3, zatem wszystko jest w porządku. Następnie Stefan sprawdza ważność czterech kleksów i tak ta runda dowodu kończy się sukcesem.  $\square$

Tak jak w poprzednio omawianych systemach dowodu Stefan uzna dowód za ważny z prawdopodobieństwem 1, mamy zatem zapewnioną zupełność. Jakie jest prawdopodobieństwo tego, że Stefan zaakceptuje dowód, gdy  $G$  nie jest 3-kolorowalny? W tym przypadku dla dowolnego kolorowania musi istnieć co najmniej jedna krawędź  $ij$ , której końce  $i$  oraz  $j$  mają ten sam kolor. Stefan ma szansę wybrać taką krawędź z prawdopodobieństwem równym co najmniej  $1/m$ , zatem prawdopodobieństwo oszukania Stefana przez Dorotę we wszystkich  $m^2$  rundach nie przekracza liczby

$$\left(1 - \frac{1}{m}\right)^{m^2}.$$

Wiadomo, że  $(1 - 1/m)^m$  dąży do  $e^{-1}$ , gdy  $m$  dąży do  $\infty$ , zatem istnieje liczba całkowita  $m_0$ , taka że  $(1 - 1/m)^m \leq 2/e$  dla  $m \geq m_0$ . Stąd  $(1 - 1/m)^{m^2} \leq (2/e)^m$  dla  $m \geq m_0$ . Funkcja  $(2/e)^m$  zmiennej  $m = |E|$  dąży do zera wykładniczo, z czego możemy wnioskować również o adekwatności.

Zajmijmy się teraz ustaleniem, czy mamy do czynienia z systemem dowodu o wiedzy zerowej. Jedyne dane, które widzi Stefan w danej rundzie dowodu, to zaszyfrowane 3-kolorowanie grafu  $G$  oraz dwa różne kolory końcowych wierzchołków pewnej krawędzi, poprzednio zaszyfrowane i wiążące dla Doroty. W każdej rundzie kolory są permutowane, wydaje się zatem, iż Stefan nie ma możliwości łączenia uzyskanych informacji w celu odtworzenia 3-kolorowania.

System dowodu nie jest systemem o wiedzy zerowej, gwarantuje jednakże słabszą wersję tej cechy, zwaną *obliczeniową wiedzą zerową*. Definiujemy ją tak jak wiedzę zerową, z tą różnicą, że odpowiednie rozkłady prawdopodobieństwa transkrypcji mogą być jedynie wielomianowo nieodróżnialne (w sensie określonym w rozdziale 12).

Zacznijmy od pokazania, jak można fałszować transkrypcje. Podamy algorytm generujący fałszywe transkrypcje, niczym nie różniące się od tych, które wytwarza uczciwy Stefan. Gdy Stefan sprzeniewierza się protokołowi, można zbudować symulator używający algorytmu  $V^*$  jako odnawialnego podprogramu do tworzenia fałszywych transkrypcji. Oba algorytmy fałszowania wzorują się na podobnych algorytmach dla systemu dowodu izomorficzności grafów.

Tu rozpatrzymy tylko pierwszy przypadek, tj. gdy Stefan wiernie realizuje protokół. Transkrypcja  $T$  w interakcyjnym dowodzie 3-kolorowalności grafu ma postać

$$(G; A_1; \dots; A_{m^2}),$$

gdzie  $A_j$  składa się z  $2n$  kleksów obliczonych przez Dorotę, krawędzi  $uv$  wybranej przez Stefana, kolorów przypisanych wierzchołkom  $u$  i  $v$  przez Dorotę w rundzie  $j$  oraz z czterech losowych liczb użytych przez Dorotę do szyfrowania kolorów tych wierzchołków. Fałszowanie transkrypcji odbywa się zgodnie z algorytmem przedstawionym na rysunku 13.13.

Dane wejściowe: graf  $G = (V, E)$  na zbiorze wierzchołków  $\{1, \dots, n\}$

- (1)  $T = (G)$
- (2) **for**  $j = 1$  **do**  $m^2$  **do**
- (3)     wybierz losowo krawędź  $\{u, v\} \in E$
- (4)     wybierz losowo dwa różne kolory  $d = d_1d_2$  i  $e = e_1e_2$ , gdzie  $d_1, d_2, e_1, e_2 \in \{0, 1\}$
- (5)     wybierz losowo element  $r_{i,j}$  ze zbioru  $X$  dla  $1 \leq i \leq n$ ,  $j = 1, 2$
- (6)     dla  $1 \leq i \leq n$  oraz  $j = 1, 2$  zdefiniuj

$$R_{i,j} = \begin{cases} f(1, r_{i,j}), & \text{gdy } i \neq u, v \\ f(d_j, r_{i,j}), & \text{gdy } i = u \\ f(e_j, r_{i,j}), & \text{gdy } i = v \end{cases}$$

- (7)     dołącz

$$(R_{1,1}, \dots, R_{n,2}, u, v, d_1, d_2, r_{d,1}, r_{d,2}, e_1, e_2, r_{e,1}, r_{e,2})$$

na koniec transkrypcji  $T$

**RYSUNEK 13.13.** Algorytm fałszowania transkrypcji dowodu 3-kolorowalności grafu

Wykazanie, że system jest systemem o (obliczeniowej) wiedzy zerowej dla Stefana, wymagałoby udowodnienia, iż oba rozkłady prawdopodobieństwa na transkrypcjach (utworzonych przez Stefana w wyniku udziału w realizacji protokołu i wyprodukowanych przez symulator) są nieodróżnialne. Nie zrobimy tego tutaj, ograniczymy się jedynie do kilku uwag. Zauważmy, że oba rozkłady prawdopodobieństwa nie są identyczne. Dzieje się tak dlatego, że niemal wszystkie wartości  $R_{ij}$  w fałszywej transkrypcji są kleksami ukrywającymi 1, podczas gdy w rzeczywistej transkrypcji są one (zazwyczaj) szyframi bardziej równomiernie rozłożonych bitów 0 i 1. Można jednak wykazać, że rozkłady te są nieodróżnialne w czasie wielomianowym, jeśli tylko użyty schemat wiążącego bitu jest bezpieczny. Dokładniej, oznacza to, że rozkład prawdopodobieństwa na kleksach

reprezentujących kolor  $c$  jest nieodróżnialny od rozkładu prawdopodobieństwa na kleksach reprezentujących kolor  $d$ , jeśli  $c \neq d$ .

Czytelnik znający teorię problemów NP-zupełnych wie, że mając dowód o wiedzy zerowej dla jednego problemu NP-zupełnego, możemy uzyskać dowód o wiedzy zerowej dla dowolnego innego takiego problemu. Wystarczy przekształcić wielomianowo ów problem NP-zupełny w problem 3-kolorowalności grafów.

---

### 13.5. Rozumowanie o wiedzy zerowej

Zreasumujmy podstawowe własności dowodu 3-kolorowalności grafu o obliczeniowej wiedzy zerowej, które zostały opisane w poprzednim podrozdziale. Żadne dodatkowe założenia nie są potrzebne do wykazania zupełności i adekwatności protokołu. Z kolei do wykazania, iż jest to dowód o wiedzy zerowej potrzebne jest pewne założenie obliczeniowe, a mianowicie bezpieczeństwo schematu wiążącego bitu. Uczestnicząc w realizacji protokołu z Dorotą, Stefan może po jego zakończeniu próbować złamać użyty w nim schemat. Na przykład, gdyby był on oparty na problemie reszt kwadratowych, Stefan mógłby szukać rozkładu modułu na dwa czynniki pierwsze. Gdyby kiedykolwiek udało mu się złamać schemat wiążącego bitu, potrafiłby odszyfrować kleksy używane przez Dorotę w ramach protokołu i dzięki temu odtworzyć 3-kolorowanie.

Analiza ta zależy od własności kleksów użytych w protokole. O ile zobowiązalność jest cechą bezwarunkową, o tyle tajność zależy od pewnych założeń obliczeniowych.

Ciekawą odmianę stanowi użycie kleksów w sytuacji, w której tajność jest bezwarunkowa, natomiast zobowiązalność wymaga założeń obliczeniowych. Prowadzi to do protokołów znanych jako *rozumowania o wiedzy zerowej* – w odróżnieniu od dowodów o wiedzy zerowej. Czytelnik z pewnością pamięta o naszym dotychczasowym założeniu o wszechmocy Doroty. Rozważając rozumowania o wiedzy zerowej, przyjmiemy, że wszystkie obliczenia Doroty muszą odbywać się w czasie wielomianowym. (W istocie nie powoduje to dodatkowych utrudnień, ponieważ, jak stwierdziliśmy wcześniej, obliczenia Doroty mieścią się w czasie wielomianowym, gdy zna ona 3-kolorowanie grafu  $G$ ).

Zacznijmy od opisu kilku tego typu schematów wiążącego bitu; potem zbadamy konsekwencje użycia takich schematów w protokole dla problemu izomorficzności grafów.

Pierwszy schemat (znów) opiera się na problemie reszt kwadratowych. Niech  $n = pq$  dla pewnych liczb pierwszych  $p$  oraz  $q$  i niech  $n \in \text{QR}(n)$  (przypomnijmy, że w poprzednim schemacie  $m$  było pseudokwadratem). Tym razem Dorota nie powinna znać ani rozkładu liczby  $n$ , ani pierwiastka kwadratowego liczby  $m$ . Dlatego albo powinien te wartości obliczyć Stefan, albo należy je uzyskać od trzeciej, zaufanej strony.

Niech  $X = \mathbb{Z}_n^*$  i  $Y = \text{QR}(n)$ . Definiujemy

$$f(b, x) = m^b x^2 \bmod n.$$

Jak poprzednio, Dorota szyfruje wartość  $b$ , wybierając losowo  $x$  i obliczając kleks  $y = f(b, x)$ . Teraz wszystkie kleksy w schemacie są resztami kwadratowymi. Dalej, każdy element  $y \in \text{QR}(n)$  jest zarazem szyfrem 0 i szyfrem 1. Istotnie, założymy, że  $y = x^2 \bmod n$  i  $m = k^2 \bmod n$ . Wtedy

$$y = f(0, x) = f(1, xk^{-1} \bmod n).$$

Oznacza to, że tajność otrzymuje się bezwarunkowo. Co się natomiast dzieje z zobowiązalnością? Dorota może odczytać dowolny kleks jednocześnie jako 0 i jako 1 wtedy i tylko wtedy, gdy potrafi obliczyć  $k$ , pierwiastek kwadratowy z liczby  $m$ . Tak więc zobowiązalność będzie zapewniona tylko wtedy, gdy przyjmiemy, że znalezienie tego pierwiastka jest dla Doroty zadaniem obliczeniowo niewykonalnym. (Gdyby Dorota pozostawała wszechmocna, mogłaby to, oczywiście, zrobić. To jeden z powodów, dla których zakładamy teraz, że ma ona ograniczone możliwości obliczeniowe).

Drugim przykładem schematu wiążącego bitu niech będzie schemat oparty na problemie logarytmu dyskretnego. Niech  $p$  będzie liczbą pierwszą, taką że problem logarytmu dyskretnego w  $\mathbb{Z}_p^*$  jest obliczeniowo nieroziwiążalny. Przyjmijmy ponadto, że  $\alpha$  jest elementem pierwotnym w  $\mathbb{Z}_p^*$  i  $\beta \in \mathbb{Z}_p^*$ . Tej ostatniej wartości, czyli  $\beta$ , nie powinna wybierać Dorota, lecz Stefan – albo może ona pochodzić od zaufanej osoby trzeciej.

W tym schemacie  $X = \{0, \dots, p-1\}$ ,  $Y = \mathbb{Z}_p^*$ , a funkcja  $f$  jest zdefiniowana następująco:

$$f(b, x) = \beta^b \alpha^x \bmod p.$$

Nietrudno stwierdzić, że schemat ten jest bezwarunkowo tajny oraz że jest zobowiązujący wtedy i tylko wtedy, gdy znalezienie logarytmu dyskretnego  $\log_\alpha \beta$  jest dla Doroty zadaniem obliczeniowo niewykonalnym.

Założymy teraz, że jeden z tych dwóch schematów wiążącego bitu stosujemy w protokole dla 3-kolorowalności grafu. Widać, że protokół zachowuje zupełność. Teraz jednak warunek adekwatności zależy od założenia obliczeniowego: protokół jest adekwatny wtedy i tylko wtedy, gdy schemat wiążącego bitu jest zobowiązujący. Czy zachowuje się także warunek zerowej wiedzy? Dzięki bezwarunkowej tajności schematu wiążącego bitu protokół jest teraz nie tylko protokołem o obliczeniowej wiedzy zerowej, ale już doskonałym protokołem o wiedzy zerowej. Mamy zatem doskonałe rozumowanie o wiedzy zerowej.

O tym, czy lepsze jest rozumowanie czy dowód, decyduje konkretne zastosowanie, podobnie jak i o tym, czy warto czynić dodatkowe założenia obliczeniowe dotyczące Doroty i Stefana. W tablicy 13.1 podsumowujemy własności dowodów i rozumowań. W kolumnie „dowód o wiedzy zerowej” założenia obliczeniowe dotyczą mocy obliczeniowej Doroty; w kolumnie „rozumowanie o wiedzy zerowej” odnoszą się one do mocy obliczeniowej Stefana.

**TABLICA 13.1.** Porównanie własności dowodów i rozumowań

| Własność                 | Dowód o wiedzy zerowej | Rozumowanie o wiedzy zerowej |
|--------------------------|------------------------|------------------------------|
| zupełność                | bezwarkunkowa          | bezwarkunkowa                |
| adekwatność              | bezwarkunkowa          | obliczeniowa                 |
| zerowa wiedza            | obliczeniowa           | doskonała                    |
| zobowiązywalność kleksów | bezwarkunkowa          | obliczeniowa                 |
| tajność kleksów          | obliczeniowa           | bezwarkunkowa                |

### 13.6. Uwagi i bibliografia

Większość materiału z tego rozdziału zaczerpnęliśmy z prac Brassarda, Chauma i Crépeau [BCC88] oraz Goldreicha, Micaliego i Wigdersona [GMW91]. Przedstawione tu schematy wiążącego bitu oraz szerokie omówienie różnic między dowodami i rozumowaniem można znaleźć w [BCC88] (należy jednak stwierdzić, że termin „rozumowanie” pojawił się po raz pierwszy w [BC90]). Dowody izomorficzności, nieizomorficzności oraz 3-kolorowalności grafów o wiedzy zerowej znajdują się w [GMW91]. Autorami kolejnej ważnej pracy są Goldwasser, Micali i Rackoff [GMR89], którzy po raz pierwszy zdefiniowali formalnie pojęcie interakcyjnego systemu dowodu. Dowód o wiedzy zerowej dla reszt kwadratowych pochodzi właśnie z tej pracy.

Blum [BL82] jest autorem pomysłu rzucania monetą przez telefon.

Bardzo nieformalną i zabawną ilustrację pojęcia wiedzy zerowej zawiera praca Quisquatera i Guillou [QG90]. Warto zajrzeć również do pracy Johnsona [JO88], która stanowi bardziej matematyczny przegląd interakcyjnych systemów dowodu.

### Ćwiczenia

- 13.1. Wykaż, że interakcyjny system dowodu dla problemu niereszt kwadratowych, przedstawiony na rysunku 13.14, jest adekwatny i zupełny oraz wyjaśnij, dlaczego protokół nie jest protokołem o wiedzy zerowej.
- 13.2. Zbuduj interakcyjny system dowodu dla problemu nieprzynależności do podgrupy. Udowodnij adekwatność i zupełność Twojego protokołu.
- 13.3. Rozważ dowód o wiedzy zerowej dla reszt kwadratowych przedstawiony na rysunku 13.8.
  - (a) Zdefiniuj ważną trójkę jako trójkę postaci  $(y, i, z)$ , gdzie  $y \in \text{QR}(n)$ ,  $i = 0$  lub  $1$ ,  $z \in \mathbb{Z}_n^*$  oraz  $z^2 \equiv x^i y \pmod{n}$ . Wykaż, że liczba ważnych trójelek jest równa  $2(p-1)(q-1)$  oraz że każda ważna trójka jest generowana z jednako-wym prawdopodobieństwem, gdy Dorota i Stefan wiernie realizują protokół.

Dane wejściowe: liczba całkowita z nieznanym rozkładem  $n = pq$ , gdzie  $p$  i  $q$  są liczbami pierwszymi, oraz  $x \in \overline{\text{QR}}(n)$

- (1) powtórz następujące kroki  $\log_2 n$  razy:
- (2) Stefan wybiera losowo  $v \in \mathbb{Z}_n^*$  i oblicza

$$y = v^2 \pmod{n}$$

następnie wybiera losowo  $i = 0$  lub  $1$  i wysyła

$$z = x^i y \pmod{n}$$

do Doroty

- (3) jeśli  $z \in \text{QR}(n)$ , to Dorota ustala  $j = 0$ , w przeciwnym razie określa  $j = 1$ , po czym wysyła  $j$  do Stefana
- (4) Stefan sprawdza, czy  $i = j$
- (5) Stefan akceptuje dowód Doroty, jeśli obliczenie w kroku 4 daje pozytywny wynik dla każdej z  $\log_2 n$  rund

**RYSUNEK 13.14.** Interakcyjny system dowodu dla problemu niereszt kwadratowych

- (b) Wykaż, że Stefan może generować trójkę o tym samym rozkładzie prawdopodobieństwa, nie znając faktoryzacji  $n = pq$ .
  - (c) Udowodnij, że protokół jest doskonałym protokołem o wiedzy zerowej dla Stefana.
- 13.4. Rozważ dowód o wiedzy zerowej dla przynależności do podgrupy przedstawiony na rysunku 13.10.
- (a) Udowodnij, że protokół jest adekwatny i zupełny.
  - (b) Zdefiniuj ważną trójkę jako trójkę postaci  $(\gamma, i, h)$ , gdzie  $\gamma \in \mathbb{Z}_n^*$ ,  $i = 0$  lub  $1$ ,  $0 \leq h \leq \ell - 1$  oraz  $\alpha^h \equiv \beta^i \gamma \pmod{n}$ . Wykaż, że liczba ważnych tróisek jest równa  $2\ell$  oraz że każda trójka jest generowana z jednakowym prawdopodobieństwem, gdy Dorota i Stefan wiernie realizują protokół.
  - (c) Wykaż, że Stefan może generować trójki o tym samym rozkładzie prawdopodobieństwa, nie znając wartości logarytmu dyskretnego  $\log_\alpha \beta$ .
  - (d) Udowodnij, że protokół jest doskonałym protokołem o wiedzy zerowej dla Stefana.
- 13.5. Udowodnij, że schemat wiążącego bitu oparty na problemie logarytmu dyskretnego przedstawiony w podrozdziale 13.5 jest bezwarunkowo tajny. Wykaż również, że jest zobowiązujący wtedy i tylko wtedy, gdy Dorota nie jest w stanie obliczyć  $\log_\alpha \beta$ .
- 13.6. Założmy, że do konstrukcji rozumowania o wiedzy zerowej dla problemu 3-kolorowalności grafu użyliśmy schematu wiążącego bitu opartego na problemie reszt kwadratowych. Udowodnij, korzystając z algorytmu fałszowania przedstawionego na rysunku 13.13, że otrzymaliśmy doskonały protokół o wiedzy zerowej dla Stefana.

---

## Dalsze lektury

---

Wśród zalecanych podręczników i monografii poświęconych kryptografii warto wymienić następujące:

- |                                     |                           |
|-------------------------------------|---------------------------|
| Beker i Piper [BP82]                | Patterson [PA87]          |
| Beutelspracher [BE94]               | Pomerance [Po90A]         |
| Brassard [BR88]                     | Rhee [RH94]               |
| Bilham i Shamir [BS93]              | Rueppel [Ru86]            |
| Denning [DE92]                      | Salomaa [SA90]            |
| Kahn [KA2004]                       | Schneier [Sc2002]         |
| Kaufman, Perlman i Speciner [KPS95] | Seberry i Pieprzyk [SP89] |
| Koblitz [KO95]                      | Simmons [Si92B]           |
| Konheim [KO81]                      | Stallings [ST97]          |
| Kranakis [KR86]                     | van Tilborg [vT88]        |
| Menezes [ME93]                      | Wayner [WA96]             |
| Meyer i Matyas [MM82]               | Welsh [WE88]              |

Pełnym i wysoce zalecanym źródłem obejmującym wszystkie aspekty praktycznej kryptografii jest książka Menezesa, Van Oorschota i Vanstone'a [MVV96].

*Journal of Cryptology, Designs, Codes and Cryptography* oraz *Cryptologia* to główne czasopisma badawcze w dziedzinie kryptologii. *Journal of Cryptology* jest czasopismem Międzynarodowego Stowarzyszenia Badań Kryptologicznych (International Association for Cryptologic Research, IACR), które sponsoruje także dwie główne doroczne konferencje kryptologiczne, CRYPTO oraz EUROCRYPT.

CRYPTO odbywa się w Santa Barbara od 1981 roku. Materiały z tych konferencji publikowane są corocznie od 1982 roku:

- |                   |                  |                  |
|-------------------|------------------|------------------|
| CRYPTO'82 [CRS83] | CRYPTO'87 [Po88] | CRYPTO'92 [BR93] |
| CRYPTO'83 [Ch84]  | CRYPTO'88 [Go90] | CRYPTO'93 [ST94] |
| CRYPTO'84 [BCS85] | CRYPTO'89 [BR90] | CRYPTO'94 [DE94] |
| CRYPTO'85 [Wi86]  | CRYPTO'90 [MV91] | CRYPTO'95 [Co95] |
| CRYPTO'86 [Od87]  | CRYPTO'91 [FE92] | CRYPTO'96 [Ko96] |

Konferencja EUROCRYPT odbywa się corocznie od 1982 roku i poza latami 1983 i 1986 publikowano jej materiały, tak jak to widać na poniższym wykazie:

|                      |                      |
|----------------------|----------------------|
| EUROCRYPT'82 [BE83]  | EUROCRYPT'91 [DA91A] |
| EUROCRYPT'84 [BCI85] | EUROCRYPT'92 [RU93]  |
| EUROCRYPT'85 [Pi86]  | EUROCRYPT'93 [HE94]  |
| EUROCRYPT'87 [CP88]  | EUROCRYPT'94 [DE95]  |
| EUROCRYPT'88 [Gu88A] | EUROCRYPT'95 [GQ95]  |
| EUROCRYPT'89 [QV90]  | EUROCRYPT'96 [MA96]  |
| EUROCRYPT'90 [DA91]  |                      |

Odbędzie się także trzecia seria konferencji, AUSCRYPT/ASIACRYPT, we współpracy z IACR. Jej materiały również były publikowane.

|                      |                     |
|----------------------|---------------------|
| AUSCRYPT'90 [SP90]   | AUSCRYPT'92 [SZ92]  |
| ASIACRYPT'91 [IRM93] | ASIACRYPT'94 [PS95] |

---

## Bibliografia

---

- [ACGS88] W. 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** (1988), 194–209.
- [AN91] H. ANTON. *Elementary Linear Algebra* (Sixth Edition). John Wiley and Sons, 1991.
- [BHS93] D. BAYER, S. HABER AND W. S. STORNETTA. Improving the efficiency and reliability of digital time-stamping. In *Sequences II, Methods in Communication, Security, and Computer Science*, pages 329–334. Springer-Verlag, 1993.
- [BB88] P. BEAUCHEMIN AND G. BRASSARD. A generalization of Hellman’s extension to Shannon’s approach to cryptography. *Journal of Cryptology*, **1** (1988), 129–131.
- [BBCGP88] P. BEAUCHEMIN, G. BRASSARD, C. CRÉPEAU, C. GOUTIER AND C. POMERANCE. The generation of random numbers that are probably prime. *Journal of Cryptology*, **1** (1988), 53–64.
- [BC94] A. BEIMEL AND B. CHOR. Interaction in key distribution schemes. *Lecture Notes in Computer Science*, **773** (1994), 444–455. (Advances in Cryptology – CRYPTO’93).
- [BP82] H. BEKER AND F. PIPER. *Cipher Systems, The Protection of Communications*. John Wiley and Sons, 1982.
- [BL90] J. BENALOH AND J. LEICHTER. Generalized secret sharing and monotone functions. *Lecture Notes in Computer Science*, **403** (1990), 27–35. (Advances in Cryptology – CRYPTO’88).
- [BE83] T. BETH (ED.). *Cryptography Proceedings*, 1982. *Lecture Notes in Computer Science*, vol. 149, Springer-Verlag, 1983.
- [BCI85] T. BETH, N. COT AND I. INGEMARSSON (EDS). *Advances in Cryptology: Proceedings of EUROCRYPT’84*. *Lecture Notes in Computer Science*, vol. 209, Springer-Verlag, 1985.
- [BJL85] T. BETH, D. JUNGNICKEL AND H. LENZ. *Design Theory*. Bibliographisches Institut, Zurich, 1985.
- [BE94] A. BEUTELSPACHER. *Cryptology*. Mathematical Association of America, 1994.

- [BS91] E. BIHAM AND A. SHAMIR. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, **4** (1991), 3–72.
- [BS93] E. BIHAM AND A. SHAMIR. *Differential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag, 1993.
- [BS93A] E. BIHAM AND A. SHAMIR. Differential cryptanalysis of the full 16-round DES. *Lecture Notes in Computer Science*, **740** (1993), 494–502. (Advances in Cryptology – CRYPTO'92).
- [BL79] G. R. BLAKLEY. Safeguarding cryptographic keys. *AFIPS Conference Proceedings*, **48** (1979), 313–317.
- [BC85] G. R. BLAKLEY AND D. CHAUM (EDS). *Advances in Cryptology: Proceedings of CRYPTO'84. Lecture Notes in Computer Science*, vol. 196, Springer-Verlag, 1985.
- [BL85] R. BLOM. An optimal class of symmetric key generation schemes. *Lecture Notes in Computer Science*, **209** (1985), 335–338. (Advances in Cryptology – EUROCRYPT'84).
- [BBS86] L. BLUM, M. BLUM AND M. SHUB. A simple unpredictable random number generator. *SIAM Journal on Computing*, **15** (1986), 364–383.
- [BL82] M. BLUM. Coin flipping by telephone: a protocol for solving impossible problems. In *24th IEEE Spring Computer Conference*, pages 133–137. IEEE Press, 1982.
- [BG85] M. BLUM AND S. GOLDWASSER. An efficient probabilistic public-key cryptosystem that hides all partial information. *Lecture Notes in Computer Science*, **196** (1985), 289–302. (Advances in Cryptology – CRYPTO'84).
- [BM84] M. BLUM AND S. MICALI. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, **13** (1984), 850–864.
- [Bo89] J. BOYAR. Inferring sequences produced by pseudo-random number generators. *Journal of Association for Computing Machinery*, **36** (1989), 129–141.
- [BDSV93] C. BLUNDO, A. DE SANTIS, D. R. STINSON AND U. VAC-CARO. Graph decompositions and secret sharing schemes. *Lecture Notes in Computer Science*, **658** (1993), 1–24. (Advances in Cryptology – EUROCRYPT'92).
- [BDSHKVY93] C. BLUNDO, A. DE SANTIS, A. HERZBERG, S. KUTTEN, U. VACCARO AND M. YUNG. Perfectly-secure key distribution for dynamic conferences. *Lecture Notes in Computer Science*, **740** (1993), 471–486. (Advances in Cryptology – CRYPTO'92).
- [BC93] J. N. E. BOS AND D. CHAUM. Probably unforgeable signatures. *Lecture Notes in Computer Science*, **740** (1993), 1–14. (Advances in Cryptology – CRYPTO'92).
- [BR88] G. BRASSARD. *Modern Cryptology – A Tutorial. Lecture Notes in Computer Science*, vol. 325, Springer-Verlag, 1988.
- [BR90] G. BRASSARD (ED.). *Advances in Cryptology – CRYPTO'89 Proceedings. Lecture Notes in Computer Science*, vol. 435, Springer-Verlag, 1990.

- [BB88A] G. BRASSARD AND P. BRATLEY. *Algorithmics, Theory and Practice*. Prentice Hall, 1988.
- [BCC88] G. BRASSARD, D. CHAUM AND C. CRÉPEAU. Minimum disclosure proofs of knowledge. *Journal of Computer and Systems Science*, **37** (1988), 156–189.
- [BC90] G. BRASSARD AND C. CRÉPEAU. Sorting out zero-knowledge. *Lecture Notes in Computer Science*, **434** (1990), 181–191. (Advances in Cryptology – EUROCRYPT’89).
- [BR89] D. M. BRESSOUD. *Factorization and Primality Testing*. Springer-Verlag, 1989.
- [BR85] E. F. BRICKELL. Breaking iterated knapsacks. *Lecture Notes in Computer Science*, **218** (1986), 342–358. (Advances in Cryptology – CRYPTO’85).
- [BR89A] E. F. BRICKELL. Some ideal secret sharing schemes. *Journal of Combinatorial Mathematics and Combinatorial Computing*, **9** (1989), 105–113.
- [BR93] E. F. BRICKELL (ED.). *Advances in Cryptology – CRYPTO’92 Proceedings*. *Lecture Notes in Computer Science*, vol. 740, Springer-Verlag, 1993.
- [BD91] E. F. BRICKELL AND D. M. DAVENPORT. On the classification of ideal secret sharing schemes. *Journal of Cryptology*, **4** (1991), 123–134.
- [BM92] E. F. BRICKELL AND K. S. MCCURLEY. An interactive identification scheme based on discrete logarithms and factoring. *Journal of Cryptology*, **5** (1992), 29–39.
- [BMP87] E. F. BRICKELL, J. H. MOORE AND M. R. PURTILL. Structure in the  $S$ -boxes of DES. *Lecture Notes in Computer Science*, **263** (1987), 3–8. (Advances in Cryptology – CRYPTO’86).
- [BO92] E. F. BRICKELL AND A. M. ODLYZKO. Cryptanalysis, a survey of recent results. In *Contemporary Cryptology, The Science of Information Integrity*, pages 501–540. IEEE Press, 1992.
- [BS92] E. F. BRICKELL AND D. R. STINSON. Some improved bounds on the information rate of perfect secret sharing schemes. *Journal of Cryptology*, **5** (1992), 153–166.
- [BKPS90] L. BROWN, M. KWAN, J. PIEPRZYK AND J. SEBERRY. LOKI – A cryptographic primitive for authentication and secrecy applications. *Lecture Notes in Computer Science*, **453** (1990), 229–236. (Advances in Cryptology – AUSCRYPT’90).
- [BDB92] M. BURMESTER, Y. DESMEDT AND T. BETH. Efficient zero-knowledge identification schemes for smart cards. *The Computer Journal*, **35** (1992), 21–29.
- [CDGV93] R. M. CAPOCELLI, A. DE SANTIS, L. GARGANO AND U. VACCARO. On the size of shares for secret sharing schemes. *Journal of Cryptology*, **6** (1993), 157–167.
- [CH95] F. CHABAUD. On the security of some cryptosystems based on error-correcting codes. *Lecture Notes in Computer Science*, to appear. (Advances in Cryptology – EUROCRYPT’94).

- [CH84] D. CHAUM (ED.). *Advances in Cryptology: Proceedings of CRYPTO'83*. Plenum Press, 1984.
- [CP88] D. CHAUM AND W. L. PRICE (EDS). *Advances in Cryptology – EUROCRYPT'87 Proceedings. Lecture Notes in Computer Science*, vol. 304, Springer-Verlag, 1988.
- [CRS83] D. CHAUM, R. L. RIVEST AND A. T. SHERMAN (EDS). *Advances in Cryptology: Proceedings of CRYPTO'82*. Plenum Press, 1983.
- [CvA90] D. CHAUM AND H. VAN ANTWERPEN. Undeniable signatures. *Lecture Notes in Computer Science*, **435** (1990), 212–216. (Advances in Cryptology – CRYPTO'89).
- [CvHP92] D. CHAUM, E. VAN HEIJST AND B. PFITZMANN. Cryptographically strong undeniable signatures, unconditionally secure for the signer. *Lecture Notes in Computer Science*, **576** (1992), 470–484. (Advances in Cryptology – CRYPTO'91).
- [CR88] B. CHOR AND R. L. RIVEST. A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE Transactions on Information Theory*, **45** (1988), 901–909.
- [Co95] D. COPPERSMITH (ED.). *Advances in Cryptology – CRYPTO'95 Proceedings. Lecture Notes in Computer Science*, vol. 963, Springer-Verlag, 1995.
- [CKM94] D. COPPERSMITH, H. KRAWCZYK AND Y. MANSOUR. The shrinking generator. *Lecture Notes in Computer Science*, **773** (1994), 22–39. (Advances in Cryptology – CRYPTO'93).
- [CSV94] D. COPPERSMITH, J. STERN AND S. VAUDENAY. Attacks on the birational permutation signature schemes. *Lecture Notes in Computer Science*, **773** (1994), 435–443. (Advances in Cryptology – CRYPTO'93).
- [CW91] T. W. CUSICK AND M. C. WOOD. The REDOC-II cryptosystem. *Lecture Notes in Computer Science*, **537** (1991), 545–563. (Advances in Cryptology – CRYPTO'90).
- [DA90] I. B. DAMGÅRD. A design principle for hash functions. *Lecture Notes in Computer Science*, **435** (1990), 416–427. (Advances in Cryptology – CRYPTO'89).
- [DA91] I. B. DAMGÅRD (ED.). *Advances in Cryptology – EUROCRYPT'90 Proceedings. Lecture Notes in Computer Science*, vol. 473, Springer-Verlag, 1991.
- [DLP93] I. DAMGÅRD, P. LANDROCK AND C. POMERANCE. Average case error estimates for the strong probable prime test. *Mathematics of Computation*, **61** (1993), 177–194.
- [DA91A] D. W. DAVIES (ED.). *Advances in Cryptology – EUROCRYPT'91 Proceedings. Lecture Notes in Computer Science*, vol. 547, Springer-Verlag, 1991.
- [DE84] J. M. DELAURENTIS. A further weakness in the common modulus protocol for the RSA cryptosystem. *Cryptologia*, **8** (1984), 253–259.

- [DBB92] B. DEN BOER AND A. BOSSALAERS. An attack on the last two rounds of MD4. *Lecture Notes in Computer Science*, **576** (1992), 194–203. (Advances in Cryptology – CRYPTO’91).
- [DE92] D. E. R. DENNING. *Kryptografia i ochrona danych*. WNT, 1992.
- [DE95] A. DE SANTIS (ED.). *Advances in Cryptology – EUROCRYPT’94 Proceedings. Lecture Notes in Computer Science*, vol. 950, Springer-Verlag, 1995.
- [DE94] Y. G. DESMEDT (ED.). *Advances in Cryptology – CRYPTO’94 Proceedings. Lecture Notes in Computer Science*, vol. 839, Springer-Verlag, 1994.
- [DWQ93] D. DE WAELFFE AND J.-J. QUISQUATER. Better login protocols for computer networks. *Lecture Notes in Computer Science*, **741** (1993), 50–70. (Computer Security and Industrial Cryptography, State of the Art and Evolution, ESAT Course, May 1991).
- [D192] W. DIFFIE. The first ten years of public-key cryptography. In *Contemporary Cryptology, The Science of Information Integrity*, pages 135–175. IEEE Press, 1992.
- [DH76] W. DIFFIE AND M. E. HELLMAN. Multiuser cryptographic techniques. *AFIPS Conference Proceedings*, **45** (1976), 109–112.
- [DH76A] W. DIFFIE AND M. E. HELLMAN. New directions in cryptography. *IEEE Transactions on Information Theory*, **22** (1976), 644–654.
- [DVW92] W. DIFFIE, P. C. VAN OORSCHOT AND M. J. WIENER. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, **2** (1992), 107–125.
- [EB93] H. EBERLE. A high-speed DES implementation for network applications. *Lecture Notes in Computer Science*, **740** (1993), 527–545. (Advances in Cryptology – CRYPTO’92).
- [EL85] T. ELGAMAL. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, **31** (1985), 469–472.
- [EAKMM86] D. ESTES, L. M. ADLEMAN, K. KOMPELLA, K. S. McCURLEY AND G. L. MILLER. Breaking the Ong-Schnorr-Shamir signature schemes for quadratic number fields. *Lecture Notes in Computer Science*, **218** (1986), 3–13. (Advances in Cryptology – CRYPTO’85).
- [FFS88] U. FEIGE, A. FIAT AND A. SHAMIR. Zero-knowledge proofs of identity. *Journal of Cryptology*, **1** (1988), 77–94.
- [FE92] J. FEIGENBAUM (ED.). *Advances in Cryptology – CRYPTO’91 Proceedings. Lecture Notes in Computer Science*, vol. 576, Springer-Verlag, 1992.
- [FE73] H. FEISTEL. Cryptography and computer privacy. *Scientific American*, **228**(5) (1973), 15–23.
- [FN91] A. FIAT AND M. NAOR. Rigorous time/space trade-offs for inverting functions. In *Proceedings of the 23rd Symposium on the Theory of Computing*, pages 534–541. ACM Press, 1991.

- [FS87] A. FIAT AND A. SHAMIR. How to prove yourself: practical solutions to identification and signature problems. *Lecture Notes in Computer Science*, **263** (1987), 186–194. (Advances in Cryptology – CRYPTO'86).
- [FOM91] A. FUJIOKA, T. OKAMOTO AND S. MIYAGUCHI. ESIGN: an efficient digital signature implementation for smart cards. *Lecture Notes in Computer Science*, **547** (1991), 446–457. (Advances in Cryptology – EUROCRYPT'91).
- [GIB91] J. K. GIBSON. Discrete logarithm hash function that is collision free and one way. *IEE Proceedings-E*, **138** (1991), 407–410.
- [GMS74] E. N. GILBERT, F. J. MACWILLIAMS AND N. J. A. SLOANE. Codes which detect deception. *Bell Systems Technical Journal*, **53** (1974), 405–424.
- [GIR91] M. GIRAUT. Self-certified public keys. *Lecture Notes in Computer Science*, **547** (1991), 490–497. (Advances in Cryptology – EUROCRYPT'91).
- [GP91] C. M. GOLDIE AND R. G. E. PINCH. *Communication Theory*. Cambridge University Press, 1991.
- [GMW91] O. GOLDREICH, S. MICALI AND A. WIGDERSON. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, **38** (1991), 691–729.
- [Go90] S. GOLDWASSER (ED.). *Advances in Cryptology – CRYPTO'88 Proceedings. Lecture Notes in Computer Science*, vol. 403, Springer-Verlag, 1990.
- [GM84] S. GOLDWASSER AND S. MICALI. Probabilistic encryption. *Journal of Computer and Systems Science*, **28** (1984), 270–299.
- [GMR89] S. GOLDWASSER, S. MICALI AND C. RACKOFF. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, **18** (1989), 186–208.
- [GMT82] S. GOLDWASSER, S. MICALI AND P. TONG. Why and how to establish a common code on a public network. In *23rd Annual Symposium on the Foundations of Computer Science*, pages 134–144. IEEE Press, 1982.
- [GM93] D. M. GORDON AND K. S. MCCURLEY. Massively parallel computation of discrete logarithms. *Lecture Notes in Computer Science*, **740** (1993), 312–323. (Advances in Cryptology – CRYPTO'92).
- [GQ88] L. C. GUILLOU AND J.-J. QUISQUATER. A practical zero-knowledge protocol fitted to security microprocessor mini-mizing both transmission and memory. *Lecture Notes in Computer Science*, **330** (1988), 123–128. (Advances in Cryptology – EUROCRYPT'88).
- [GQ95] L. C. GUILLOU AND J.-J. QUISQUATER (EDS). *Advances in Cryptology – EUROCRYPT'95 Proceedings. Lecture Notes in Computer Science*, vol. 921, Springer-Verlag, 1995.
- [Gu88] C. G. GUNTHER Alternating step generators controlled by de Bruijn sequences. *Lecture Notes in Computer Science*, **304** (1988), 88–92. (Advances in Cryptology – EUROCRYPT'87).

- [GU88A] C. G. GUNTHER (ED.). *Advances in Cryptology – EUROCRYPT’88 Proceedings. Lecture Notes in Computer Science*, vol. 330, Springer-Verlag, 1988.
- [HS91] S. HABER AND W. S. STORNETTA. How to timestamp a digital document. *Journal of Cryptology*, **3** (1991), 99–111.
- [HSS93] J. HÄSTAD, A. W. SCHRIFT AND A. SHAMIR. The discrete logarithm modulo a composite hides  $O(n)$  bits. *Journal of Computer and Systems Science*, **47** (1993), 376–404.
- [HE80] M. E. HELLMAN. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, **26** (1980), 401–406.
- [Hi29] L. S. HILL. Cryptography in an algebraic alphabet. *American Mathematical Monthly*, **36** (1929), 306–312.
- [HE94] T. HELLESETH (ED.). *Advances in Cryptology – EUROCRYPT’93 Proceedings. Lecture Notes in Computer Science*, vol. 765, Springer-Verlag, 1994.
- [HLLPRW91] D. G. HOFFMAN, D. A. LEONARD, C. C. LINDNER, K. T. PHELPS, C. A. RODGER AND J. R. WALL. *Coding Theory, The Essentials*. Marcel Dekker, 1991.
- [IRM93] H. IMAI, R. L. RIVEST AND T. MATSUMOTO (EDS). *Advances in Cryptology – ASIACRYPT’91 Proceedings. Lecture Notes in Computer Science*, vol. 739, Springer-Verlag, 1993.
- [ISN87] M. ITO, A. SAITO AND T. NISHIZEKI. Secret sharing scheme realizing general access structure. *Proceedings IEEE Globecom’87*, pages 99–102, 1987.
- [Jo88] D. S. JOHNSON. The NP-completeness column: an ongoing guide. *Journal of Algorithms*, **9** (1988), 426–444.
- [KA2004] D. KAHN. *Lamacze kodów. Historia kryptologii*. WNT, 2004.
- [KPS95] C. KAUFMAN, R. PERLMAN AND M. SPECINER. *Network Security. Private Communication in a Public World*. Prentice Hall, 1995.
- [Ko87] N. KOBLITZ. Elliptic curve cryptosystems. *Mathematics of Computation*, **48** (1987), 203–209.
- [Ko95] N. KOBLITZ. *Wykład z teorii liczb i kryptografii*. WNT, 1995.
- [Ko96] N. KOBLITZ (ED.). *Advances in Cryptology – CRYPTO’96 Proceedings. Lecture Notes in Computer Science*, vol. 1109, Springer-Verlag, 1996.
- [KN93] J. KOHL AND C. NEUMAN. *The Kerberos Network Authentication Service*. Network Working Group Request for Comments: 1510, September 1993.
- [Ko81] A. G. KONHEIM. *Cryptography, A Primer*. John Wiley and Sons, 1981.
- [Kr86] E. KRANAKIS. *Primality and Cryptography*. John Wiley and Sons, 1986.
- [LA90] J. C. LAGARIAS. Pseudo-random number generators in cryptography and number theory. In *Cryptology and Computational Number Theory*, pages 115–143. American Mathematical Society, 1990.

- [LO91] B. A. LAMACCHIA AND A. M. ODLYZKO. Computation of discrete logarithms in prime fields. *Designs, Codes and Cryptography*, **1** (1991), 47–62.
- [LL93] A. K. LENSTRA AND H. W. LENSTRA, JR. (EDS). *The Development of the Number Field Sieve. Lecture Notes in Mathematics*, vol. 1554. Springer-Verlag, 1993.
- [LL90] A. K. LENSTRA AND H. W. LENSTRA, JR. Algorithms in number theory. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 673–715. Elsevier Science Publishers, 1990.
- [LN83] R. LIDL AND H. NIEDERREITER. *Finite Fields*. Addison-Wesley, 1983.
- [LW88] D. L. LONG AND A. WIDGERSON. The discrete log hides  $O(\log n)$  bits. *SIAM Journal on Computing*, **17** (1988), 363–372.
- [MS77] F. J. MACWILLIAMS AND N. J. A. SLOANE. *The Theory of Error-Correcting Codes*. North-Holland, 1977.
- [MA86] J. L. MASSEY. Cryptography – a selective survey. In *Digital Communications*, pages 3–21. North-Holland, 1986.
- [MA94] M. MATSUI. Linear cryptanalysis method for DES cipher. *Lecture Notes in Computer Science*, **765** (1994), 386–397. (Advances in Cryptology – EUROCRYPT’93).
- [MA94A] M. MATSUI. The first experimental cryptanalysis of the data encryption standard. *Lecture Notes in Computer Science*, **839** (1994), 1–11. (Advances in Cryptology – CRYPTO’94).
- [MTI86] T. MATSUMOTO, Y. TAKASHIMA AND H. IMAI. On seeking smart public-key distribution systems. *Transactions of the IECE (Japan)*, **69** (1986), 99–106.
- [MA96] U. MAURER (ED.). *Advances in Cryptology – EUROCRYPT’96 Proceedings. Lecture Notes in Computer Science*, vol. 1070, Springer-Verlag, 1996.
- [MC90] K. MCCURLEY. The discrete logarithm problem. In *Cryptology and Computational Number Theory*, pages 49–74. American Mathematical Society, 1990.
- [MC78] R. McELIECE. A public-key cryptosystem based on algebraic coding theory. *DSN Progress Report*, **42–44** (1978), 114–116.
- [MC87] R. McELIECE. *Finite Fields for Computer Scientists and Engineers*. Kluwer Academic Publishers, 1987.
- [ME93] A. J. MENEZES. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.
- [MBGMVY93] A. J. MENEZES, I. F. BLAKE, X. GAO, R. C. MULLIN, S. A. VANSTONE AND T. YAGHOOBIAN. *Applications of Finite Fields*. Kluwer Academic Publishers, 1993.
- [MOV94] A. J. MENEZES, T. OKAMOTO AND S. A. VANSTONE. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, **39** (1993), 1639–1646.

- [MV91] A. J. MENEZES AND S. A. VANSTONE (EDS). *Advances in Cryptology – CRYPTO'90 Proceedings. Lecture Notes in Computer Science*, vol. 537, Springer-Verlag, 1991.
- [MV93] A. J. MENEZES AND S. A. VANSTONE. Elliptic curve cryptosystems and their implementation. *Journal of Cryptology*, **6** (1993), 209–224.
- [MVV2004] A. J. MENEZES, P. C. VAN OORSCHOT AND S. A. VANSTONE. *Kryptografia stosowana*. WNT, 2004.
- [ME78] R. C. MERKLE. Secure communications over insecure channels. *Communications of the ACM*, **21** (1978), 294–299.
- [ME90] R. C. MERKLE. One way hash functions and DES. *Lecture Notes in Computer Science*, **435** (1990), 428–446. (Advances in Cryptology – CRYPTO'89).
- [ME90A] R. C. MERKLE. A fast software one-way hash function. *Journal of Cryptology*, **3** (1990), 43–58.
- [MH78] R. C. MERKLE AND M. E. HELLMAN. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, **24** (1978), 525–530.
- [MM82] C. MEYER AND S. MATYAS. *Cryptography: A New Dimension in Computer Security*. John Wiley and Sons, 1982.
- [Mi76] G. L. MILLER. Riemann's hypothesis and tests for primality. *Journal of Computer and Systems Science*, **13** (1976), 300–317.
- [Mi86] V. MILLER. Uses of elliptic curves in cryptography. *Lecture Notes in Computer Science*, **218** (1986), 417–426. (Advances in Cryptology – CRYPTO'85).
- [MPW92] C. J. MITCHELL, F. PIPER AND P. WILD. Digital signatures. In *Contemporary Cryptology, The Science of Information Integrity*, pages 325–378. IEEE Press, 1992.
- [Mi91] S. MIYAGUCHI. The FEAL cipher family. *Lecture Notes in Computer Science*, **537** (1991), 627–638. (Advances in Cryptology – CRYPTO'90).
- [MOI90] S. MIYAGUCHI, K. OHTA AND M. IWATA. 128-bit hash function ( $N$ -hash). *Proceedings of SECURICOM 1990*, 127–137.
- [Mo92] J. H. MOORE. Protocol failures in cryptosystems. In *Contemporary Cryptology, The Science of Information Integrity*, pages 541–558. IEEE Press, 1992.
- [NBS77] *Data Encryption Standard (DES)*. National Bureau of Standards FIPS Publication 46, 1977.
- [NBS80] *DES modes of operation*. National Bureau of Standards FIPS Publication 81, 1980.
- [NBS81] *Guidelines for implementing and using the NBS data encryption standard*. National Bureau of Standards FIPS Publication 74, 1981.
- [NBS85] *Computer data authentication*. National Bureau of Standards HPS Publication 113, 1985.
- [NBS93] *Secure hash standard*. National Bureau of Standards FIPS Publication 180, 1993.

- [NBS94] *Digital signature standard*. National Bureau of Standards FIPS Publication 186, 1994.
- [OD87] A. M. ODLYZKO (ED.). *Advances in Cryptology – CRYPTO’86 Proceedings. Lecture Notes in Computer Science*, vol. 263, Springer-Verlag, 1987.
- [OK93] T. OKAMOTO. Provably secure and practical identification schemes and corresponding signature schemes. *Lecture Notes in Computer Science*, **740** (1993), 31–53. (Advances in Cryptology – CRYPTO’92).
- [OSS85] H. ONG, C. P. SCHNORR AND A. SHAMIR. Efficient signature schemes based on polynomial equations. *Lecture Notes in Computer Science*, **196** (1985), 37–46. (Advances in Cryptology – CRYPTO’84).
- [PA87] W. PATTERSON. *Mathematical Cryptology for Computer Scientists and Mathematicians*. Rowman and Littlefield, 1987.
- [PE86] R. PERALTA. Simultaneous security of bits in the discrete log. *Lecture Notes in Computer Science*, **219** (1986), 62–72. (Advances in Cryptology – EUROCRYPT’85).
- [Pi86] F. PICHLER (ED.). *Advances in Cryptology – EUROCRYPT’85 Proceedings. Lecture Notes in Computer Science*, vol. 219, Springer-Verlag, 1986.
- [PS95] J. PIEPRZYK AND R. SAFAVI-NAINI (EDS). *Advances in Cryptology – ASIACRYPT’94 Proceedings. Lecture Notes in Computer Science*, vol. 917, Springer-Verlag, 1995.
- [PB45] R. L. PLACKETT AND J. P. BURMAN. The design of optimum multi-factorial experiments. *Biometrika*, **33** (1945), 305–325.
- [PH78] S. C. POHLIG AND M. E. HELLMAN. An improved algorithm for computing logarithms over  $GF(p)$  and its cryptographic significance. *IEEE Transactions on Information Theory*, **24** (1978), 106–110.
- [Po88] C. POMERANCE (ED.). *Advances in Cryptology – CRYPTO’87 Proceedings. Lecture Notes in Computer Science*, vol. 293, Springer-Verlag, 1988.
- [Po90] C. POMERANCE. Factoring. In *Cryptology and Computational Number Theory*, pages 27–47. American Mathematical Society, 1990.
- [Po90A] C. POMERANCE (ED.). *Cryptology and Computational Number Theory*, American Mathematical Society, 1990.
- [PGV93] B. PRENEEL, R. GOYAERTS AND J. VANDEWALLE. Information authentication: hash functions and digital signatures. *Lecture Notes in Computer Science*, **741** (1993), 87–131. (Computer Security and Industrial Cryptography, State of the Art and Evolution, ESAT Course, May 1991).
- [PGV94] B. PRENEEL, R. GOYAERTS AND J. VANDEWALLE. Hash functions based on block ciphers: a synthetic approach. *Lecture Notes in Computer Science*, **773** (1994), 368–378. (Advances in Cryptology – CRYPTO’93).
- [QG90] J.-J. QUISQUATER AND L. GUILLOU. How to explain zero-knowledge protocols to your children. *Lecture Notes in Computer Science*, **435** (1990), 628–631. (Advances in Cryptology – CRYPTO’89).

- [QV90] J.-J. QUISQUATER AND J. VANDEWALLE (EDS). *Advances in Cryptology – EUROCRYPT’89 Proceedings. Lecture Notes in Computer Science*, vol. 434, Springer-Verlag, 1990.
- [RA79] M. O. RABIN. Digitized signatures and public-key functions as intractable as factorization. *MIT Laboratory for Computer Science Technical Report*, LCS/TR-212, 1979.
- [RA80] M. O. RABIN. Probabilistic algorithms for testing primality. *Journal of Number Theory*, **12** (1980), 128–138.
- [RH94] M. Y. RHEE. *Cryptography and Secure Communications*. McGraw-Hill, 1994.
- [Ri91] R. L. RIVEST. The MD4 message digest algorithm. *Lecture Notes in Computer Science*, **537** (1991), 303–311. (Advances in Cryptology – CRYPTO’90).
- [RSA78] R. L. RIVEST, A. SHAMIR AND L. ADLEMAN. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, **21** (1978), 120–126.
- [Ro93] K. H. ROSEN. *Elementary Number Theory and its Applications* (Third Edition). Addison Wesley, 1993.
- [Ru86] R. A. RUEPPEL. *Analysis and Design of Stream Ciphers*. Springer-Verlag, 1986.
- [Ru93] R. A. RUEPPEL (ED.). *Advances in Cryptology – EUROCRYPT’92 Proceedings. Lecture Notes in Computer Science*, vol. 658, Springer-Verlag, 1993.
- [RV94] R. A. RUEPPEL AND P. C. VAN OORSCHOT. Modern key agreement techniques. To appear in *Computer Communications*, 1994.
- [Sa90] A. SALOMAA. *Public-Key Cryptography*. Springer-Verlag, 1990.
- [Sc94] J. I. SCHILLER. Secure distributed computing. *Scientific American*, **271**(5) (1994), 72–76.
- [Sc2002] B. SCHNEIER. *Kryptografia dla praktyków. Protokoły, algorytmy i programy źródłowe w języku C*. WNT, 2002.
- [Sc91] C. P. SCHNORR. Efficient signature generation by smart cards. *Journal of Cryptology*, **4** (1991), 161–174.
- [SP89] J. SEBERRY AND J. PIEPRZYK. *Cryptography: An Introduction to Computer Security*. Prentice-Hall, 1989.
- [SP90] J. SEBERRY AND J. PIEPRZYK (EDS). *Advances in Cryptology – AUSCRYPT’90 Proceedings. Lecture Notes in Computer Science*, vol. 453, Springer-Verlag, 1990.
- [SZ92] J. SEBERRY AND Y. ZHENG (EDS). *Advances in Cryptology – AUSCRYPT’92 Proceedings. Lecture Notes in Computer Science*, vol. 718, Springer-Verlag, 1993.
- [Sh79] A. SHAMIR. How to share a secret. *Communications of the ACM*, **22** (1979), 612–613.
- [Sh84] A. SHAMIR. A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem. *IEEE Transactions on Information Theory*, **30** (1984), 699–704.

- [SH90] A. SHAMIR. An efficient identification scheme based on permuted kernels. *Lecture Notes in Computer Science*, **435** (1990), 606–609. (Advances in Cryptology – CRYPTO'89).
- [SH94] A. SHAMIR. Efficient signature schemes based on birational permutations. *Lecture Notes in Computer Science*, **773** (1994), 1–12. (Advances in Cryptology – CRYPTO'93).
- [SH48] C. E. SHANNON. A mathematical theory of communication. *Bell Systems Technical Journal*, **27** (1948), 379–423, 623–656.
- [SH49] C. E. SHANNON. Communication theory of secrecy systems. *Bell Systems Technical Journal*, **28** (1949), 656–715.
- [ST92] J. H. SILVERMAN AND J. TATE. *Rational Points on Elliptic Curves*. Springer-Verlag, 1992.
- [Si85] G. J. SIMMONS. Authentication theory/coding theory. *Lecture Notes in Computer Science*, **196** (1985), 411–432. (Advances in Cryptology – CRYPTO'84).
- [Si88] G. J. SIMMONS. A natural taxonomy for digital information authentication schemes. *Lecture Notes in Computer Science*, **293** (1988), 269–288. (Advances in Cryptology – CRYPTO'87).
- [Si92] G. J. SIMMONS. A survey of information authentication. In *Contemporary Cryptology, The Science of Information Integrity*, pages 379–419. IEEE Press, 1992.
- [Si92A] G. J. SIMMONS. An introduction to shared secret and/or shared control schemes and their application. In *Contemporary Cryptology, The Science of Information Integrity*, pages 441–497. IEEE Press, 1992.
- [Si92B] G. J. SIMMONS (ED.). *Contemporary Cryptology, The Science of Information Integrity*. IEEE Press, 1992.
- [SB92] M. E. SMID AND D. K. BRANSTAD. The data encryption standard: past and future. In *Contemporary Cryptology, The Science of Information Integrity*, pages 43–64. IEEE Press, 1992.
- [SB93] M. E. SMID AND D. K. BRANSTAD. Response to comments on the NIST proposed digital signature standard. *Lecture Notes in Computer Science*, **740** (1993), 76–88. (Advances in Cryptology – CRYPTO'92).
- [SS77] R. SOLOVAY AND V. STRASSEN. A fast Monte Carlo test for primality. *SIAM Journal on Computing*, **6** (1977), 84–85.
- [ST97] W. STALLINGS. *Ochrona danych w sieci i intersieci. W teorii i w praktyce*. WNT, 1997.
- [ST88] D. R. STINSON. Some constructions and bounds for authentication codes. *Journal of Cryptology*, **1** (1988), 37–51.
- [ST90] D. R. STINSON. The combinatorics of authentication and secrecy codes. *Journal of Cryptology*, **2** (1990), 23–49.
- [ST92] D. R. STINSON. Combinatorial characterizations of authentication codes. *Designs, Codes and Cryptography*, **2** (1992), 175–187.
- [ST92A] D. R. STINSON. An explication of secret sharing schemes. *Designs, Codes and Cryptography*, **2** (1992), 357–390.

- [St94] D. R. STINSON (ED.). *Advances in Cryptology – CRYPTO’93 Proceedings. Lecture Notes in Computer Science*, vol. 773, Springer-Verlag, 1994.
- [vHP93] E. VAN HEYST AND T. P. PEDERSEN. How to make efficient fail-stop signatures. *Lecture Notes in Computer Science*, **658** (1993), 366–377. (Advances in Cryptology – EUROCRYPT’92).
- [VV89] S. A. VANSTONE AND P. C. VAN OORSCHOT. *An Introduction to Error Correcting Codes with Applications*. Kluwer Academic Publishers, 1989.
- [vT88] H. C. A. VAN TILBORG. *An Introduction to Cryptology*. Kluwer Academic Publishers, 1988.
- [vT93] J. VAN TILBURG. Secret-key exchange with authentication. *Lecture Notes in Computer Science*, **741** (1993), 71–86. (Computer Security and Industrial Cryptography, State of the Art and Evolution, ESAT Course, May 1991).
- [VV84] U. VAZIRANI AND V. VAZIRANI. Efficient and secure pseudorandom number generation. In *Proceedings of the 25th Annual Symposium on the Foundations of Computer Science*, pages 458–463. IEEE Press, 1984.
- [WA90] M. WALKER. Information-theoretic bounds for authentication systems. *Journal of Cryptology*, **2** (1990), 131–143.
- [WA96] P. WAYNER. *Disappearing Cryptography*. Academic Press, 1996.
- [WE88] D. WELSH. *Codes and Cryptography*. Oxford Science Publications, 1988.
- [Wi94] M. J. WIENER. Efficient DES key search. Technical report TR-244, School of Computer Science, Carleton University, Ottawa, Canada, May 1994 (also presented at CRYPTO’93 Rump Session).
- [Wi80] H. C. WILLIAMS. A modification of the RSA public-key encryption procedure. *IEEE Transactions on Information Theory*, **26** (1980), 726–729.
- [Wi86] H. C. WILLIAMS (ED.). *Advances in Cryptology – CRYPTO’85 Proceedings. Lecture Notes in Computer Science*, vol. 218, Springer-Verlag, 1986.
- [YA82] A. YAO. Theory and applications of trapdoor functions. In *Proceedings of the 23rd Annual Symposium on the Foundations of Computer Science*, pages 80–91. IEEE Press, 1982.

---

# *Skorowidz*

---

$\varepsilon$ -algorytm przewidywania poprzedniego bitu 375  
– rozróżniający 374  
( $k, l$ )-PRGB 360, *zob.* też generatorów binarnych pseudolosowych

## A

adekwatność 388, 389, 398, 405, 407, 409, 410  
Adleman L. 116, 157  
Agencja Bezpieczeństwa Narodowego 84  
Alexi W. 384  
alfabet binarny 22  
– klucza strumieniowego 21  
algorytm deterministyczny 131  
– Dixon'a 154  
– „dowodu fałszerstwa” 226  
– dzielenia próbnego 152  
– Euklidesa 118–120, 122, 126, 127, 141, 154, 158, 180, 183, 299  
– – – rozszerzony 119–121, 126, 184, 201  
– fałszowania 392, 393, 395, 405, 406, 410  
– dla problemu 3-kolorowalności grafu 405, 406, 410  
– – – izomorficzności grafu 393, 395  
– Huffmana 56, 69  
– krzywych eliptycznych 152, 154, 156, 157  
– Las Vegas 140, 141, 149, 163, 173  
– – –, probabilistyczny 237, 239  
– Lenstry 154  
– Millera–Rabina 130, 131, 137, 138, 139

algorytm Millera–Rabina, prawdopodobieństwo błędu 139  
– Monte Carlo 130, 140, 375, 376, 402  
– – negatywnie nastawiony 131, 375  
– – neutralny 375, 376, 378  
– – pozytywnie nastawiony 131  
– – prawdopodobieństwo błędu 131, 376, 377, 378, 384, 385  
– – neutralny 375  
–  $p - 1$  Williamsa 152  
– – Pollarda 152, 163  
– podpisu 205, 220, 226, 255, 256, 266, 273, 286, 301  
– – TSS 256  
– Pohliga–Hellmania 168–172, 185, 189, 200, 201  
– probabilistyczny 131  
– programowania całkowitoliczbowego Lenstry 195  
– przewidywania następnego bitu 367, 368, 369, 370, 378, 384  
– – poprzedniego bitu 374, 375  
– rozkładu na czynniki pierwsze, *zob.* algorytmy faktoryzacyjne  
– rozróżniający 368, 369, 370, 375, 378  
– Schoofa 188  
– Shanksa 167–168, 185, 189, 201, 232  
– sita kwadratowego 152, 156, 157  
– Solovaya–Strassena 130, 131, 134, 136, 137  
– ułamkówłańcuchowych 152  
– weryfikacji 204, 205, 207, 215, 226, 266, 286  
– znajdowania kluczów 114

algorytmy faktoryzacji 152-157  
 alternatywa 252  
 – wykluczająca 22, 85, 91  
 Anton H. 40  
 arytmetyka reszt 3  
 atak metodą dnia urodzin 239-241, 249  
 – Pohliga–Hellmana 189  
 – przez podstawienie 306, 307, 308  
 – – podszywanie się 306, 307  
 – różnicowy, zob. kryptoanaliza różnicowa  
 – typu intruz–w–środku 272, 273, 274, 275, 277, 280, 306  
 – z powtórzeniem 271  
 – z wybranym tekstem jawnym 25, 89, 95, 113, 151  
 – – – zaszyfrowanym 25, 151  
 – ze znanyim tekstem jawnym 25, 37, 38, 52, 84  
 – – – zaszyfrowanym 46, 52, 62  
 Atkins D. 157  
 autentyczność 87, 88

**B**

Bayer D. 257  
 baza 332, 346, 347, 352, 353, 354, 355, 356, 359  
 – rozkładu 154, 155, 172  
 Beauchemin P. 69  
 Beimel A. 281  
 Beker H. 26, 40  
 Benaloh J. 333, 357  
 Beth T. 303, 324  
 bezpieczeństwo algorytmu DES 84  
 – bezwarunkowe 46, 222, 229, 261, 262, 264, 305, 332  
 – – tajnego schematu współużytkowania 332-333  
 – doskonale 52  
 – funkcji skrótu 239, 241, 245, 253  
 – generatora bitów pseudolosowych 365  
 – – Bluma–Bluma–Shuba 373-378  
 – obliczeniowe 45, 52, 117, 205, 305  
 – protokołu 289  
 – schematu podpisu 208, 210, 212, 222, 226, 229, 236, 277  
 – systemu kryptograficznego 7, 9, 25, 45, 84, 189, 267, 315

bezpieczny standard skrótu 250, 253, 257, 301  
 Biham E. 91  
 bit parzystości 77  
 Blakley G. R. 357  
 blok podstawień 74, 75, 81-83, 84, 85, 92, 93, 96, 97  
 Blom R. 281  
 Blum M. 380, 384, 409  
 Blundo C. 281, 357  
 boczne wyjście 118, 127  
 Bos J. N. E. 231  
 Bosselaers A. 257  
 Boyar J. 383  
 Brandstad D. K. 112  
 Brassard G. 69, 158, 384, 409  
 Bratley P. 158, 384  
 Brent R. 157  
 Bressoud D. M. 158  
 Brickell E. F. 112, 201, 302, 357, 358  
 Burman J. P. 324  
 Burmester M. 303

**C**

Capocelli R. M. 358  
 CBC 86, 87, 113  
 certyfikat 266, 276, 287, 288, 289, 291, 292, 297  
 CFB 86, 87, 88, 113  
 Chabaud F. 201  
 Chaum D. 231, 232, 257, 409  
 Chor B. 201, 384  
 ciało 11, 41  
 – Galois 182-185  
 ciąg bitów pseudolosowych 360, 361, 362, 378, 380  
 – superrosnący 193, 194  
 – wag 193, 194, 203  
 Coppersmith D. 232, 363, 383  
 Crépeau C. 409

**D**

Damgard I. 257  
 datowanie 255, 257  
 datownik 255, 256, 269, 271  
 Davenport D. M. 358  
 Davis J. A. 157  
 de Waleffe D. 303

dekodowanie syndromu 197  
 den Boer B. 257  
 Denning D. 40  
 DES 52, 66, 72-115, 116, 130, 249, 251,  
 260, 285  
 –, 1-rundowa charakterystyka 101  
 –, 3-rundowa charakterystyka 102, 111,  
 114  
 –, 3-rundowy 93-100  
 –, 4-rundowy 115  
 –, 6-rundowy 100-111  
 –, bezpieczeństwo 84  
 –, blok podstawień 74, 75, 81-83, 84,  
 85, 92, 93, 96, 97  
 –, funkcja rozszerzania 73, 75, 85, 96  
 –, kompromis dotyczący wymagań  
 czas-pamięć 89-91  
 –, kryptoanaliza 91-112  
 –, macierze liczników 98, 99, 105  
 –,  $n$ -rundowa charakterystyka 100  
 –,  $n$ -rundowy 100  
 –, opis 72-80  
 –, permutacja początkowa 72-73, 75,  
 91  
 –, S-blok 74, 81, 84, 85, 92, 93, 96, 97  
 –, tryb CBC 86  
 –, – CFB 86  
 –, – ECB 86  
 –, – OFB 86, 363  
 –, tryby 86-89  
 –, wejściowa różnica symetryczna 92,  
 93, 94, 101, 102  
 –, wektor początkowy 86, 87  
 –, wyjściowa różnica symetryczna 92,  
 93, 95, 96, 97, 101, 102  
 –, zestaw podkluczny 73, 77, 78, 80, 99  
 Desmedt Y. G. 303  
 Diffie W. 85, 116, 157, 231, 273, 281  
 Digital Equipment Corporation 85  
 digram 42, 62, 63  
 długość krytyczna 65, 70  
 dodawanie, element neutralny 3  
 –, – przeciwny 3  
 –, liczb całkowitych modulo 252  
 –, modularne 291  
 domknięcie 333, 353  
 dopełnienie 252

doskonały tajny schemat współużytkowania 332, 333, 334, 337, 339, 341,  
 342, 349, 351, 354, 355  
 dowody o wiedzy zerowej 387-410  
 – – – dla problemów NP-zupełnych  
 400  
 dowód fałszerstwa 230-231  
 – o obliczeniowej wiedzy zerowej 400,  
 402-407  
 – przez wiedzę 288  
 DSS, zob. standard podpisu cyfrowego  
 dysjunktywna postać normalna 334,  
 342, 343  
 dystrybucja klucza 260-283  
 – – przez TA 261  
 dzielenie próbne 152

## E

ECB 86, 113  
 element neutralny 3  
 – odwrotny 10, 11  
 – pierwotny 184, 208  
 – – modulo  $n$  124, 125  
 – przeciwny 3  
 ElGamal T. 165  
 entropia 52-60, 321-323, 324, 325, 349,  
 350, 357  
 – języka 63  
 – kodu uwierzytelniania 321-323  
 – tajnego schematu współużytkowania  
 349-352  
 – warunkowa 60, 61, 322  
 –, własności 57-60  
 Estes D. 232  
 etykieta uwierzytelniająca 306, 307,  
 315, 322

## F

faktoryzacja 128, 152-157, 158, 253,  
 268, 364  
 –, algorytmy 152-158  
 –, baza rozkładu 154  
 –, dzielenie próbne 152  
 –, metoda  $p - 1$  152  
 –, – sita kwadratowego 154-156  
 –, przez pocztę elektroniczną 157  
 fałszerstwo 236  
 FEAL 112

- Fiat A. 12, 303  
 Friedman William (Wolfe) 31  
 funkcja afiniczna 8  
 – boolowska 252  
 – deszyfrująca 2, 5, 7, 11, 22  
 – Eulera 10  
 – haszująca, *zob.* funkcja skrótu  
 – jednokierunkowa 117, 118, 215, 216,  
     217  
 – – z bocznym wejściem 117, 118  
 – *N*-hash 257  
 – redukcji 90  
 – rozszerzania 73, 75, 85, 254, 259  
 – skrótu 206, 235-259, 266, 286, 292,  
     297, 301  
 – –, atak metodą dnia urodzin 239-240  
 – –, bezkonfliktowa 236  
 – –, bezpieczeństwo 239, 241, 245, 253  
 – – Chauma-van Heijsta-Pfitzmann  
     241-244, 259  
 – –, jednokierunkowa 237  
 – –, kryptograficzna 235  
 – – MD4 250-255  
 – – –, bezpieczeństwo 253  
 – – –, rejesty 251  
 – – –, słowo 250, 251  
 – – MD5 250, 253  
 – – –, funkcja rozszerzania 254, 259  
 – – –, słowo 253  
 – –, obliczeniowo bezpieczna 241  
 – –, silnie bezkonfliktowa 236, 237, 244,  
     246, 248, 249, 259  
 – –, słabo bezkonfliktowa 236  
 – Snerfu 257  
 – szyfrująca 2, 5, 7, 22, 86, 127  
 – RSA 364  
 – ściągająca 235  
 – wkleśla 57  
 funkcje afiniczne 8

**G**

- generator BBS, *zob.* generator Bluma–Bluma–Shuba  
 – bitów pseudolosowych 360, 361-364,  
     366, 367, 369, 370, 378, 381, 383,  
     384  
 – – –, bezpieczeństwo 365, 378  
 – Bluma–Bluma–Shuba 371-378, 384  
 – – –, bezpieczeństwo 373-378

- generator Bluma–Bluma–Shuba,  $\varepsilon$ -od-  
     różnialny 378  
 – klucz strumieniowego 21  
 – kongruencyjny, liniowy 361-362, 367,  
     383, 384  
 – liczb pseudolosowych 364  
 – obcinający 363, 383  
 – oparty na logarytmie dyskretnym 385  
 – RSA 363-364, 384  
 Gibson J. K. 257  
 Gilbert E. N. 324  
 Girault M. 278, 281  
 Goldie C. M. 69  
 Goldreich O. 384, 409  
 Goldwasser S. 378, 380, 409  
 Gordon D. M. 200  
 Govaerts R. 257  
 graf 388, 390, 392, 393, 395, 396, 397  
 – izomorficzny 388  
 – pełny 346  
 – podgraf indukowany 353  
 – skierowany 114  
 – składowe 346  
 – spójny 352, 353  
 – wielodzielny, pełny 346, 347, 353  
 –, właściwe 3-kolorowanie 402-403  
 Graff M. 157  
 grupa 4  
 – abelowa 4, 119, 186  
 – ciała Galois, multiplikatywna 181  
 – cykliczna 124, 179, 184, 185, 187  
 – krzywej eliptycznej nad ciałem skoń-  
     czonym 181  
 – multiplikatywna 124, 185  
 – – niezerowych wielomianów 184  
 –, rzad elementu 124  
 Guillou L. C. 297, 409  
 Gunther C. G. 383

**H**

- Haber S. 257  
 Hastad J. 384  
 haszowanie 91  
 Hellman M. E. 85, 112, 116, 157, 231  
 Hill Lester S. 14  
 Hoffman D. G. 201  
 Holdridge D. B. 157

**I**

- iloczyn macierzy 15
  - skalarny 319, 344, 345
- Imai H. 275, 281
- indeks zgodności 33
  - , wzajemny 36, 44
- indukcja matematyczna 177
- integralność 87
- interpolacja wielomianowa 327
- Ito M. 357

**J**

- Johnson D. S. 409
- Jungnickel D. 324

**K**

- Kahn D. 40
- kanał komunikacyjny 1, 2, 116, 260
  - , bezpieczny 260, 261
- Kasiski Friedrich 31
- kleks 400, 401, 405, 406, 407, 408
- klucz 1, 2
  - , dystrybucja 260
  - , publiczny, samopotwierdzający 278, 279
  - samodualny 113
  - samopotwierdzający 278, 280, 281
  - sesji 262, 269, 271, 272
  - strumieniowy 20-25, 38, 39, 42, 43, 86, 361, 380, 382, 383
  - , uzgadnianie 260-283
- klucze błędne 62, 63, 65
  - dualne 113
  - równoważne 226
- Koblitz N. 158, 200, 201
- kod BCH 203
  - Hamminga 199
  - korekcji błędów 196
  - uwierzytelniania komunikatów 87, 88, 89, 113
- kodowania Huffmmana 54-57, 63
- kodowanie zmiennej 54, 55, 57
  - , długość 69
  - , skuteczność 56
  - , wolne od przedrostków 55, 56, 69

- kody Goppa 196, 198
  - uwierzytelniania wiadomości 305-325
  - , atak przez podstawienie 305, 307-309
  - , bezpieczeństwo 312
  - , charakteryzacje 320-321
  - , ograniczenia kombinatoryczne 312-315
  - , – związane z entropią 321-323
  - , podszywanie się 305, 306-308
  - , prawdopodobieństwo oszustwa 305, 306-313, 320-321
- kombinacje liniowe 14
- konfiguracje przekątniowe 314
- konflikt 239, 240, 241, 243, 246, 248, 253
- kongruencja 8, 9, 10, 11
- koniunkcja 252
- koniunktywna postać normalna 334
- konstrukcja Brickella 343
  - oparta na obwodzie monotonicznym 333-335, 337, 343, 353-354, 358
  - – przestrzeni liniowej 343, 347, 357, 359
  - przez rozkład 353-357, 359
- Kranakis E. 158, 383
- Krawczyk H. 363, 383
- K*-rozkład, idealny 354, 355
- kryptoolanaliza 5, 25-39, 44
  - liniowa 112
  - różnicowa 91-112
  - , operacja filtrowania 104
  - , para błędna 103, 104, 111
  - , – poprawna 103, 104, 105, 111
- kryptografia z kluczem publicznym 116-163, 164-203, 260
- kryptogram 1
- kryterium Eulera 132, 149, 177
- krzywe eliptyczne 185-192, 200, 203
  - supersośliwe 189
- kwadrat łaciński rzędu  $n$  69
- kwadraty łacińskie wzajemnie ortogonalne 314

**L**

- Lagarias J. C. 383
- LaMacchia B. A. 200
- Leichter J. 333, 357
- Lenstra A. K. 154, 157, 158

Lenstra H. W. 154, 157, 158  
 Lenz H. 324  
 Leyland P. 157  
 LFSR 23, 38-39, 361, 383  
 liczba pierwsza 10, 130  
 – pseudolosowa Eulera 134, 162  
 – RSA-*d* 157  
 – zredukowana 3  
 liczby Fibonacciego 130  
 – względnie pierwsze 10  
 LOKI 112  
 Long D. L. 384  
 LUCIFER 72

**M**

MAC 87, 88, 89, 113, 305  
 macierz generująca 197  
 – inwolutywna 41  
 – jednostkowa 15, 20  
 – odwrocalna 41, 42  
 – odwrotna 15, 16, 17, 20  
 – permutacji 20  
 – sprawdzania parzystości 196  
 – sprzężona 17  
 – szyfrowania 15, 41, 42, 48, 70  
 – uwierzytelniania 307, 308, 309, 310,  
     314, 315, 321  
 – Vandermonde'a 329  
 macierze liczników 98, 99  
 – ortogonalne 314, 316-317, 318, 319,  
     320, 321, 323, 324, 325  
 MacWilliams F. J. 201, 324  
 Manasse M. S. 157  
 Mansour Y. 363, 383  
 Massey J. L. 324  
 maszyna Wienera 112  
 Matsui M. 112  
 Matsumoto T. 275, 281  
 McCurley K. 302  
 McEliece R. 198  
 Menezes A. J. 189, 190, 200, 201  
 Merkle R. C. 257, 281  
 metoda  $\rho$  152  
 – ataku, *zob.* atak  
 – eliminacji Gaussa 156  
 – najbliższego sąsiada 196, 197  
 – o wiedzy zerowej 302  
 – obliczania indeksu 185, 189

metoda „podnieś do kwadratu i wyjmuj” 128, 129, 133, 159, 165, 201  
 – Shanksa, *zob.* algorytm Shanksa  
 – sita kwadratowego Pomerance'a 156  
 – syndromu 203  
 – tylko tekst zaszyfrowany 25, 26  
 – wiążącego bitu 400, *zob.* też schemat wiążącego bitu  
 – wyczerpującego przeszukiwania 5, 8,  
     13-14, 43, 89, 111, 115  
 metody ataku 25  
 miara informacji schematu, względna  
     342, 358  
 – uczestnika, względna 342, 349, 352,  
     355, 358  
 Micali S. 378, 384, 409  
 Miller V. 200  
 Mitchell C. J. 231  
 mnożenie, element neutralny 4  
 – modularne 128, 129, 154, 291  
 moduł 3  
 Moore J. H. 112, 158

**N**

nadmiarowość języka 63, 65, 70  
 Naor M. 112  
 Narodowe Biuro Standaryzacji 72  
*n*-gram 62, 63  
 niejednoznaczność klucza 60-61  
 nienaruszalność 305  
 niepodrabialność podpisu 226, 229  
 niepowodzenie protokołu 159, 210  
 – wspólnego modułu 160, 161  
 niereszta kwadratowa 132, 133, 175,  
     410  
 nierozkładalność 183  
 nierówność Jensena 57, 58, 59, 64, 317,  
     322  
 niezależność pary od pary 295  
 Nishizeki T. 357  
 NIST 207, 214, 215, 231  
 NP-zupełność 45  
*n*-rundowa charakterystyka 100  
 NSA 84, 214

**O**

obciążalność wejściowa 333  
 – wyjściowa 333

obiekt bitowy 400  
 obliczanie indeksu 172  
 obliczeniowa wiedza zerowa 405  
 obwód monotoniczny 333, 335, 354  
 odległość Hamminga 196  
 – jednostkowa 65  
 – kodu 196  
 Odlyzko A. M. 200, 201  
 odwracalność macierzy 16  
 OFB 86, 87, 113, 363  
 Okamoto T. 189, 201, 291  
 okres ważności 269, 271  
 operacja filtrowania 103, 104, 105, 111  
 osoba dowodząca 387

**P**

para błędna 103, 104, 111  
 – poprawna 103, 104, 105, 111  
 paradoks dnia urodzin 239, 241  
 parametr bezpieczeństwa 286, 297, 379  
 parametry losowości 379  
 Pedersen T. P. 226, 232  
 Peralta R. 200  
 permutacja odwrotna 7, 19, 20  
 – poczatkowa 72-73, 75, 91  
 permutacje 7, 19, 20  
 Pfitzmann B. 257  
 pierścień 183  
 – wielomianów modulo  $f(x)$  182  
 pierwiastek kwadratowy, główny 373,  
 375, 382  
 – –, nietrywialny 141  
 – –, trywialny 141  
 pierwiastki pierwotne 159  
 Pinch R. G. E. 69  
 Piper F. 26, 40, 231  
 Plackett R. L. 324  
 podgraf indukowany 352  
 podpis cyfrowy 204-234, *zob.* też schemat podpisu  
 – jednorazowy 215-226  
 – niepodrabialny 226-231  
 – niezaprzeczalny 220-226  
 podstawienie 306, 307  
 podszywanie się 306, 307  
 podzbiór nieupoważniony, minimalny  
 332  
 – upoważniony 331, 332, 338, 339, 340,  
 345

podzbiór upoważniony, baza 332  
 – –, minimalny 332  
 Pomerance C. 158  
 Porta Giovanni 18  
 postać normalna, dysjunktywna 334,  
 342, 343, 354  
 potęgowanie modularne 128, 154, 291  
 potwierdzenie klucza 271, 275  
 poufność 87, 88  
 prawdopodobieństwo *a posteriori* 48  
 – *a priori* 47, 48  
 – charakterystyki 100  
 – łączne 46  
 – oszustwa 307-311, 312, 320, 321  
 – warunkowe 46, 47, 48  
 prawo kwadratowej wzajemności 135  
 Preneel B. 257  
 PRGB, *zob.* generator bitów pseudolosowych  
 problem 3-kolorowalności grafu 402-  
 407, 409, 410  
 – decyzyjny 131, 192, 387, 388, 392, 394  
 – dekodowania 198  
 – – ogólnego kodu korekcji błędów 195  
 – Diffiego-Hellmana 267, 268, 269, 277,  
 282  
 – *i*-tego bitu 174-175  
 – izomorficzności grafów 388, 389, 390,  
 392, 393, 394, 395, 398, 399, 407,  
 409  
 – logarytmu dyskretnego 164-179, 180,  
 181, 185, 188, 189, 191, 200, 201,  
 208, 209, 210, 212, 213, 221, 227,  
 241, 253, 261, 265, 267, 268, 278,  
 280, 286, 297, 303, 364, 384, 399,  
 401, 402, 408, 410  
 – – –, bezpieczeństwo bitów 174-179  
 – – –, problem *i*-tego bitu 174-175  
 – – –, – Pohliga-Hellmana 168-172  
 – – –, uogólniony 179-181  
 – – –, w ciałach Galois 182  
 – – –, w krzywych eliptycznych 189  
 – MTI 283  
 –  $n^2$  261  
 – nieizomorficzności grafów 388, 397,  
 409  
 – nieprzynależności do podgrupy 409  
 – niereszt kwadratowych 409

- problem NP-zupełny 45, 117, 193, 195, 198, 388, 400, 402, 407
  - pakowania 117
  - plecakowy 192
  - przynależności do podgrupy 399, 410
  - reszt kwadratowych 372, 384, 401, 407, 410
  - rozkładalność 131
  - sumy podzbioru 192, 193, 195
  - wyszukiwania 193
- problemy wielomianowo równoważne 128
- procedura pytania-odpowiedź 288
- produkt 66
- progowa struktura dostępu 333
- protoły dwuprzebiegowe 275
- protokół, bezpieczeństwo 289
  - identyfikacji 289, 290, 295, 301
  - o wiedzy zerowej, doskonały 410
  - przekazania klucza sesji 270
  - typu pytanie-odpowiedź 220-226, 285, 286, 387
  - uzgadniania klucza 261, 262, 263, 271
    - – –, Diffiego–Hellmana 262, 263
    - – –, Merkle'a 262
    - – –, schemat Giraulta 262, 278-280, 283
    - – – Matsumoto–Takashimy–Imaiego 262, 275-278, 282
    - – – MTI 262, 275-278, 282
  - weryfikacji 220, 223, 289, 294
  - wydawania kluczy sesji 269
  - wymiany klucza Diffiego–Hellmana 265, 271-274
  - – – z uwierzytelnianiem 274, 275
  - wyrzeczenia 220, 223-224
  - zupełny 288, 290, 292, 299
- przeciwnik aktywny 260, 267, 270, 305
  - bierny 260, 267, 277, 282
- przekształcenie liniowe 14
  - modularne 194
- przestrzeń klucz 1, 7, 21, 66, 84, 205, 312, 379
  - , przeszukiwanie wyczerpujące 5, 8, 13-14, 35, 43, 85
- przesunięcie 3, 5
  - cykliczne 252
  - względne 34, 35

- przeszukiwanie binarne 91
  - wyczerpujące 5, 8, 13-14
- pseudokwadraty modulo  $n$  372, 375, 380
- pseudoreszty kwadratowe modulo  $n$  374
- punkt w nieskończoności 185
- Purtill M. R. 112

## Q

- Quisquater J.-J. 297, 303, 409

## R

- Rabin M. O. 157
- Rackoff C. 409
- REDOC-II 112
- redukcje modularne 135
- reguła deszyfrowania 1, 21, 66, 116, 379
  - dystrybucji 338-341, 344, 345, 349, 350, 351, 353, 354
  - szyfrowania 1, 2, 21, 50, 66, 116, 379
  - uwierzytelniania 306, 315
- Rejestr Federalny 72
- rejestr przesuwający ze sprzężeniem zwrotnym, liniowy 23, 38, 39, 361, 363, 365, 383
- rekurencja liniowa 22, 42
  - stopnia  $n$  22
- reprezentacja binarna 128
  - grupy 179, 181
- reszta kwadratowa 132, 133, 175, 186, 303, 371, 372, 374, 375, 378, 380, 397, 398, 409
- Rivest R. L. 116, 157, 201, 250, 257
- rodzina podzbiorów monotoniczna 334
- Rosen K. H. 40
- rozdzający 326, 327, 334, 338, 356, 357, 358
- rozkład na czynniki pierwsze 128, 152, 158
  - prawdopodobieństwa 47, 48, 49, 51, 52, 53, 54, 55, 56, 58, 59, 62, 63, 66, 69, 364
  - – –,  $\epsilon$ -rozróżnialne 365, 366
  - – – łącznego 59
  - – –, nierozróżnialny 393
  - – –, rozróżnialny 365, 393
  - – – warunkowego 60

rozkładalność 131  
 rozróżnialność rozkładów prawdopodobieństwa 365  
 rozumowanie o wiedzy zerowej 407-409,  
 410  
 — — —, doskonałe 409  
 różnica symetryczna 22, 51, 52, 73, 85,  
 91, 252, 361, 380  
 Rueppel R. A. 40, 282  
 rząd elementu 124  
 — macierzy 229

**S**

Saito A. 357  
 Salomaa A. 158  
 S-blok 74, 75, 81-83, 84, 85, 92, 93, 96,  
 97  
 schemat Giraulta 262, 278-280, 283  
 — identyfikacji 273, 284-304  
 — —, bezpieczeństwo 290, 292-296  
 — —, bezpieczny 284  
 — — Feigego-Fiata-Shamira 302  
 — — Guillou-Quisquatera 297-301, 302,  
 304  
 — — oparty na tożsamości 302  
 — — —, solidność 299  
 — — —, zupełność 299  
 — — Okamoto 291-297, 302, 304  
 — — —, bezpieczeństwo 292-296  
 — — oparty na tożsamości 301, 303  
 — — Schnorra 286, 287, 291, 292, 293,  
 297, 301, 304  
 — — permutacji jądra 302  
 — — permutacyjny Shamira, dwuwymier-  
 ny 232  
 — — podpisu 204-234, 237, 276, 286, 292,  
 297, 301, 303, 305  
 — — —, bezpieczeństwo 208, 210, 212, 222,  
 226, 229, 233, 236, 277  
 — — Bosa-Chauma 217-219, 233  
 — — —, definicja 205  
 — — ElGamala 207-211, 212, 231, 232,  
 233  
 — — —, bezpieczeństwo 208, 210, 212, 233  
 — — —, złamanie 210  
 — — jednorazowego 215-226  
 — — Lamperta 215-216, 219, 231, 233  
 — — —, modyfikacja Bosa-Chauma 217  
 — — niepodrabialnego 226-231

schemat podpisu niepodrabialnego,  
 bezpieczeństwo 226, 229  
 — — — van Heysta-Pedersena 226-227,  
 233, 234  
 — — niezaprzeczalnego 220-226  
 — — — Chauma-van Antwerpena 220-  
 221, 233  
 — — —, bezpieczeństwo 222  
 — — —, protokół weryfikacji 223  
 — — —, — wyzroczenia 220, 223-224  
 — — —, obliczeniowo bezpieczny 205  
 — — Ong-Schnorra-Shamira 232  
 — — RSA 206  
 — — Schnorra 303  
 — — z algorytmem weryfikacji 273, 276  
 — — — progowy 326, 330-331, 332, 337, 338,  
 342, 343, 357  
 — — — Brickella 344  
 — — — idealny 343, 345, 346, 347, 349, 354,  
 359  
 — — — Shamira 327-331, 332, 342, 343,  
 346, 358  
 — — uzgadniania klucza 260, 282  
 — — wiążącego bitu 400-402, 406, 410  
 — — —, tajny 400, 401, 402, 407, 408  
 — — —, zobowiązujący 400, 401, 402, 408,  
 410  
 — — współużytkowania 326-359  
 — — —, bezpieczeństwo bezwarunkowe 332-  
 333  
 — — —, doskonały tajny 332, 333, 334, 337,  
 339, 341, 342, 349, 351, 354, 355  
 — — —, tajny, metody geometryczne 358  
 — — — wstępnej dystrybucji klucza 261, 262-  
 269, 271, 282  
 — — — —, bezpieczeństwo 262  
 — — — — Bloma 261, 262-265, 282  
 — — — — —, bezpieczeństwo 262, 264  
 — — — — Diffiego-Hellmana 265-269, 272  
 — — wymiany klucza Diffiego-Hellmana  
 272, 275, 282  
 Schiller J. I. 281  
 Schnorr C. P. 384  
 Schrift A. W. 384  
 serwer kluczny 261, 271  
 Shamir A. 91, 116, 157, 195, 201, 302,  
 303, 357, 384  
 Shannon C. 45, 50, 52, 65, 68, 69  
 SHS, zob. bezpieczny standard skrótu

- sieci 314  
 Silverman J. H. 201  
 Simmons G. J. 157, 160, 324, 358  
 sito ciała liczbowego 152, 156, 157  
 – kwadratowe 152, 156, 157  
 składowe 346  
 skrót wiadomości 235, 239, 241, 251,  
     252, 253  
 skuteczność kodowania 56  
 Sloane N. J. A. 201, 324  
 słowo 250  
 – kluczowe 12, 13, 44  
 Smid M. E. 112  
 sprawdzający 387, 394  
 sprzężenie zwrotne, liniowe 24  
 stan źródłowy 306, 312, 315, 321-323  
 standard podpisu cyfrowego 207, 212-  
     215, 235, 241  
 – szyfrowania danych 52, 72-115  
 Stern J. 232  
 Stinson D. R. 324  
 stopień wielomianu 182  
 Stornetta W. S. 257  
 strategia najbliższego sąsiada 196, 197  
 struktura dostępu 331, 332, 333, 336,  
     337, 339, 343, 344, 346, 347, 349,  
     350, 351, 352, 353, 354, 355, 356,  
     357, 359  
 –, monotoniczna 342  
 –, progowa 332, 333  
 –, warunek monotoniczności 332  
 – progowa 332  
 strumień tekstu jawnego 20-25  
 suma docelowa 192  
 – modulo 2 22, 51  
 symbol Jacobiego 133, 134, 135, 162,  
     371, 382  
 – Legendre'a 133  
 – Newtona 32  
 symulator 392, 393, 394, 395, 396, 397,  
     399, 405, 406  
 syndrom 196, 197  
 system „dowodliwie bezpieczny” 45  
 – bezwarunkowo bezpieczny 46, 52  
 – Chora-Rivesta 117  
 – doskonale tajny 48  
 – dostarczania klucza 269  
 – dowodu, interakcyjny 387-400, 403,  
     409  
 system dowodu, interakcyjny, dla pro-  
     blemu 3-kolorowalności grafu 402-  
     409  
 – –, –, – izomorficzności grafu 388,  
     390-396  
 – –, –, – nieizomorficzności grafu 388-  
     390  
 – –, –, doskonały 398  
 – – o obliczeniowej wiedzy zerowej 400,  
     405-407, 409  
 – – –, adekwatny 388, 389, 398, 405,  
     407, 408, 409, 410  
 – – –, doskonały 392, 393, 394, 395,  
     396, 397, 398, 399, 400, 409  
 – – –, interakcyjny 398, 403, 409  
 – – –, zupełny 388, 389, 391, 398,  
     405, 407, 408, 409, 410  
 – kryptograficzny 1, 5  
 – –, bezpieczeństwo 7, 84, 315  
 – – Bluma-Goldwassera 381-383  
 – – ElGamala 117, 164-166, 179, 188,  
     189, 190, 200, 201, 202, 268, 269  
 – – –, na krzywych eliptycznych 189-190  
 – – –, uogólniony 180  
 – –, endomorficzny 66, 68  
 – – heksadecymalny 81  
 – – idempotentny 68  
 – –, iterowany 68  
 – –, nieidempotentny 68  
 – – Kerberos 262, 269-271, 275  
 – – –, klucz sesji 269, 270  
 – – –, okres ważności kluczy 270  
 – – –, znacznik czasu (datownik) 270  
 – – McEliece'a 117, 164, 195-200, 201  
 – – Menezesa-Vanstone'a 190, 191, 203  
 – –, monoalfabetyczny 12  
 – – obliczeniowo bezpieczny 45, 205  
 – – oparty na ciele skończonym 179  
 – – – krzywych eliptycznych 117, 179,  
     190, 191, 200  
 – – plecakowy Merkle'a-Hellmana 117,  
     164, 192-195, 201, 203  
 – – polialfabetyczny 14  
 – – produktowy 66, 68, 71, 74  
 – – Rabina 147-152, 157, 163  
 – – –, bezpieczeństwo 150, 151  
 – – RSA 116, 117, 118, 124, 125-127,  
     130, 139, 146, 154, 157, 159, 160,  
     205, 215, 268, 297, 364

- system kryptografii RSA, bezpieczeństwo 117, 127, 128, 297
- – –, implementacja 127-130
- – z kluczem prywatnym 116, 260, 305
- – – publicznym 26, 116-163, 164-203, 260, 379
- – – Bluma-Goldwassera 380-381, 384, 385
- – – Goldwassera-Micaliego 380, 401
- – –, parametr bezpieczeństwa 379
- – –, probabilistyczny 379, 380, 385, 401
- szesnastkowy 81
- wieloalfabetowy 14
- szłyf afiniczny 8, 9, 10, 11, 12, 40, 66, 67, 68, 69, 70
- –, kryptoanaliza 27-28
- blokowy 20, 21
- Cezara 5
- Hilla 14-18, 20, 40, 68, 70, 84
- –, afiniczny 41
- –, kryptoanaliza 37, 42
- monoalfabetyczny 32
- multiplikatywny 66, 67
- „nie do złamania” 50
- permutujący 18-20, 68
- podstawieniowy 7, 18, 40, 46, 65, 68, 70
- –, bezpieczeństwo 46
- –, kryptoanaliza 28-31
- –  $m$ -gramowy 70
- produktowy 67
- przestawieniowy 18
- strumieniowy 20-25, 43, 361
- –, kryptoanaliza 38
- –, niesynchroniczny 24
- –, okresowy 21
- –, synchroniczny 21
- Vernama 51
- Vigenére'a 12-14, 21, 22, 23, 24, 40, 43, 68, 70, 71
- –, bezpieczeństwo 46
- –, kryptoanaliza 31-36
- z autokluczem 24-25, 43
- z automatycznym generowaniem klucza 24-25, 43
- z kluczem jednorazowym 51, 52, 360-361

- szłyf z przesunięciem 3-4, 5, 8, 12, 46, 49, 50, 62, 66, 68, 70
- – –, tajność doskonała 49
- szłyfowanie dwukrotne 113
- probabilistyczne 378, 379, 384, 402
- świeżość klucza 269, 272

## T

- TA, zob. wiarygodny czynnik
- tajność doskonała 48, 49, 50, 51, 69, 70, 114, 360, 361
- tajny schemat współużytkowania, zob. schemat współużytkowania
- Takashima Y. 275, 281
- Tate J. 201
- tekst jawny 1, 21
- –, stały 162
- zaszyfrowany 1, 21
- teoria konfiguracji kombinatorycznych 314
- test Kasiskiego 31, 32, 33
- liczb pseudopierwszych, mocny 137
- pierwszości Millera-Rabina 138, 158
- – Solovaya-Strassena 134, 138, 158
- – –, prawdopodobieństwo błędu 162, 163
- uniwersalny 369
- test<sub>j</sub>* 94, 96, 97, 103, 104, 105, 115
- transkrypcja 391, 392, 394, 395, 396, 397, 405, 406

- tryb elektronicznej książki kodowej 86
- szłyfowego sprzężenia zwrotnego 86
- wiązania bloków szłyfowych 86
- wyjściowego sprzężenia zwrotnego 86
- TSS, zob. usługi datowania
- twierdzenie Bayesa 46, 47, 49, 50, 51, 61, 136
- Fermata 139, 153
- Hassego 188
- Lagrange'a 124, 162
- Lamé'go 130
- o liczbach pierwszych 130
- o resztach, chińskie 121-123, 141, 143, 144, 145, 149, 150, 158, 159, 162, 168, 172, 381, 382

## U

- udział 326, 336, 338, 340, 344, 354, 356, 357, 358

usługa datowania 256  
 uwierzytelnianie klucza, pośrednie 278,  
   280  
 uwierzytelne uzgadnianie klucza  
   273, 274  
 uzgadnianie klucza 260-283

**V**

van Antwerpen H. 232  
 van Heijst E. 257  
 van Heyst E. 226, 232  
 van Oorschot P. C. 201, 273, 281, 282  
 van Tilburg J. 282  
 Vandewalle J. 257  
 Vanstone S. A. 189, 190, 201  
 Vaudenay S. 232  
 Vaziriani U. 384  
 Vaziriani V. 384  
 Vernam G. 51, 52  
 Vigencre Blaise de 12

**W**

wagi 192  
 Walker M. 324  
 wariacje z powtórzeniami 55  
 warunek monotoniczności 332  
 ważne trójki 393, 395, 409, 410  
 wektor błędów 197  
   – początkowy 23, 86, 87  
 Welsh D. 69  
 wiadomość podpisana 306  
 wiarygodny czynnik 260  
 wielomian nierozkładalny 183, 184,  
   185, 201  
   –, stopień 182  
 Wiener M. J. 273, 281  
 Wigderson A. 384, 409  
 Wild P. 231  
 Williams H. C. 158  
 własność Spernera 218  
 wniosek Fermata 124  
 współczynnik dwumianowy 32

wstępna dystrybucja klucza 261-268  
 wykładownik deszyfrowania 278  
   – szyfrowania 278, 297  
 wymiana klucza z uwierzytelnianiem  
   273, 274  
 wyrocznia 140, 146, 176, 177  
 wyznacznik macierzy 17  
 wzór Bayesa, 325, 340, *zob. też* twier-  
   dzenie Bayesa  
   – interpolacyjny Lagrange'a 329, 358  
   – Stirlinga 70, 219

**Y**

Yao A. 384

**Z**

zalążek 21, 360, 361, 362, 366, 372, 380,  
   383  
 zależność liniowa wektorów 156  
 zasada Kerckhoffa 25  
 zbiór dopuszczalny 105, 111  
   – kluczy 327, 332, 334, 355  
   – reszt modulo  $n$  118  
   – udziałów 327, 332  
   – ważnych trójkę 393  
 zdarzenia niezależne 47  
 zestaw podkluczy 73, 77, 78, 80, 99  
 zmienna losowa 46, 52, 53, 54, 58, 63,  
   323, 350  
 zmienne losowe, niezależne 46, 47, 58,  
   60  
 znacznik czasu 255  
 znajdowanie konfliktów 239  
 zobowiązalność 400, 401, 402, 407,  
   408, 410  
 $\mathbb{Z}_p$ -rozkład, idealny 356  
 zupełność 388, 389, 391, 398, 405, 407,  
   409, 410

**Ż**

źródło bez pamięci 55