

niezbędne założenia dla większości kryptografii klucza prywatnego pokazano w [93]. Dowód Twierdzenia 7.29 pochodzi z [72].

Na naszą prezentację duży wpływ miała książka Goldreicha [75], którą gorąco polecamy osobom zainteresowanym bardziej szczegółowym zgłębieniem tematów poruszonych w tym rozdziale.

Ćwiczenia

7.1 Udowodnij, że jeśli istnieje funkcja jednokierunkowa, to istnieje funkcja jednokierunkowa f taka, że $f(0n) = 0n$ dla każdego n . Zauważ, że dla nieskończego wielu wartości y łatwo jest obliczyć f dykt jednokierunkowość? $^{-1}(y)$. Dlaczego to nie jest sprzeczne

7.2 Udowodnij, że jeśli f jest funkcją jednokierunkową, to funkcja g określona jest przez $= g(x_1, x_2) \stackrel{\text{def}}{=} f(x_1), x_2)$, gdzie $|x_1| = |x_2|$, jest także funkcją jednokierunkową. Zauważ, że g ujawnia połowę swojego wkładu, ale mimo to jest jednokierunkowe.

7.3 Udowodnij, że jeśli istnieje funkcja jednokierunkowa, to istnieje funkcja jednokierunkowa zachowująca długość.

Wskazówka: Niech f będzie funkcją jednokierunkową, a $p(\cdot)$ będzie wielomianem takim, że $|f(x)| = p(|x|)$. (Uzasadnij istnienie takiego p). Zdefiniuj $f(x) f(x)10p(|x|) - |f(x)|$. Następnie zmodyfikuj f , aby uzyskać funkcję zachowującą długość, która pozostaje jednokierunkowa.

7.4 Niech (Gen, H) będzie odporną na kolizje funkcją skrótu, gdzie H odwzorowuje ciągi o długości $2n$ na ciągi o długości n . Udowodnić, że rodzina funkcji $(\text{Gen}, \text{Samp}, H)$ jest jednokierunkowa (por. Definicja 7.3), gdzie Samp jest trywialnym algorytmem próbującym jednolity串 znaków o długości $2n$.

Podpowiedź: Wybór uniforma $x \in \{0, 1\}^{2n}$ i znalezienie odwrotności $y = H_s(x)$ nie gwarantuje kolizji. Ale najczęściej kończy się to kolizją...

7.5 Niech F będzie (zachowującą długość) permutacją pseudolosową.

(a) Pokaż, że funkcja $f(x, y) = Fx(y)$ nie jest jednokierunkowa. (b) Pokaż, że funkcja $f(y) = F0n(y)$ (gdzie $n = |y|$) nie jest jedno-

sposób.

(c) Udowodnij, że funkcja $f(x) = Fx(0n)$ (gdzie $n = |x|$) jest jednokierunkowa.

7.6 Niech f będzie funkcją jednokierunkową zachowującą długość i niech hc będzie twardym predykatem f . Zdefiniuj G jako $G(x) = f(x)hc(x)$. Czy G jest koniecznie generatorem pseudolosowym? Udowodnij swoją odpowiedź.

7.7 Udowodnić, że istnieją funkcje jednokierunkowe wtedy i tylko wtedy, gdy istnieją rodziny funkcji jednokierunkowych. Przedyskutuj, dlaczego Twój dowód nie ma zastosowania w przypadku permutacji jednokierunkowych.

7.8 Niech f będzie funkcją jednokierunkową. Czy $g(x) = f(f(x))$ koniecznie jednokierunkowe = funkcjonować? A co z $g(x) = f(x)f(f(x))$? Udowodnij swoje odpowiedzi.

7.9 Niech $\Pi = (\text{Gen}, \text{Samp}, f)$ będzie rodziną funkcji. Funkcja $hc : \{0, 1\}^{\{0, 1\}}$ jest twardym predykatem Π , jeśli można go efektywnie obliczyć i jeśli dla każdego algorytmu $ppt A$ istnieje pomijalna funkcja negl taka, że

$$\Pr_{I \sim \text{Gen}(1^n), x \in \text{Samp}(I)} [A(I, f_I(x)) = hc(I, x)] \leq \frac{1}{2} + \text{negl}(n).$$

Udowodnij wersję twierdzenia Goldreicha-Levina dla tego ustwienia, mianowicie, jeśli istnieje rodzina funkcji jednokierunkowych (lub permutacja) Π , to istnieje rodzina funkcji jednokierunkowych (lub permutacja) Π i twardy -core predykat hc .

7.10 Pokaż konstrukcję generatora pseudolosowego z dowolnej rodziny permutacji jednokierunkowych. Możesz wykorzystać wynik poprzedniego ćwiczenia.

7.11 To ćwiczenie jest przeznaczone dla studentów, którzy ukończyli kurs z teorii złożoności lub w inny sposób są zaznajomieni z zupełnością NP.

(a) Pokaż, że z istnienia funkcji jednokierunkowych implikuje się $P = N P$. (b) Załóż, że $P = N P$. Pokaż, że istnieje funkcja f , która jest: (1) obliczalna w czasie wielomianowym, (2) trudna odwrócić w najgorszym przypadku (tj. dla wszystkich probabilistycznych wielomianów-czasów A , $\Pr_{x \in \{0,1\}^n} [f(A(f(x))) = f(x)] = 1$), ale (3) nie jest jednokierunkowy.

7.12 Niech $x \in \{0, 1\}^n$ i oznaczamy $x = x_1 \cdots x_n$. Udowodnić, że jeśli istnieje funkcja jednokierunkowa, to istnieje funkcja jednokierunkowa f taka, że dla każdego i istnieje algorytm A_i taki, że

$$\Pr_{x \in \{0, 1\}^n} \left[\frac{1}{2} \sum_{i=1}^n \delta_{A_i(f(x)_i)} \right] = \frac{1}{2}.$$

(To ćwiczenie pokazuje, że nie można twierdzić, że każda funkcja jednokierunkowa ukrywa co najmniej jeden określony bit danych wejściowych.)

7.13 Pokaż, że jeśli funkcja jeden do jednego f ma twardy predykat, to f jest jednokierunkowy.

7.14 Pokaż, że jeśli Konstrukcja 7.21 zostanie zmodyfikowana w naturalny sposób tak, że $F_k(x)$ zostanie zdefiniowana dla każdego nieupustego łańcucha x o długości co najwyżej n , to konstrukcja nie będzie już funkcją pseudolosową.

7.15 Udowodnić, że jeśli istnieje funkcja pseudolosowa, która przy użyciu klucza o długości n odwzorowuje n -bitowe dane wejściowe na jednobitowe wyniki, to istnieje funkcja pseudolosowa, która odwzorowuje n -bitowe dane wejściowe na n -bitowe wyniki.

Wskazówka: Użyj klucza długości w^2 , i udowodnij, że Twoja konstrukcja jest bezpieczna, używając argumencie hybrydowym.

7.16 Udowodnić, że dwuokrągła sieć Feistela wykorzystująca pseudolosowe funkcje okrągłe (jak w równaniu (7.15)) nie jest pseudolosowa.

7.17 Udowodnić, że trójokrągła sieć Feistela wykorzystująca pseudolosowe funkcje okrągłe (jak w równaniu (7.16)) nie jest silnie pseudolosowa.

Wskazówka: jest to znacznie trudniejsze niż poprzednie ćwiczenie. Użyj wyróżnika, który tworzy dwa zapytania do permutacji i jedno zapytanie do jego odwrotności.

7.18 Rozważmy permutację z kluczem F zdefiniowaną przez

$$F_k(x) \stackrel{\text{def}}{=} \text{Feistel}_{F_k}(x).$$

(Zauważ, że w każdej rundzie używany jest ten sam klucz.) Pokaż, że F jest nie jest pseudolosowe.

7.19 Niech G będzie generatorem pseudolosowym ze współczynnikiem rozwinięcia $(n) = n+1$. Udowodnić, że G jest funkcją jednokierunkową.

7.20 Niech X, Y, Z będą zbiorami prawdopodobieństwa. Udowodnić, że jeśli $X \subseteq Y \subseteq Z$, wtedy $X \subseteq Z$.

7.21 Dowód twierdzenia 7.32.

7.22 Niech $X = \{X_n\}_n$ i $Y = \{Y_n\}_n$ będą nieroróżnialnymi obliczeniowo zespołami prawdopodobieństwa. Udowodnić, że dla dowolnego probabilistycznego algorytmu wielomianowego A zespoły $\{A(X_n)\}_n$ i $\{A(Y_n)\}_n$ są obliczeniowo nieroróżnialne.

Machine Translated by Google

Część III

Klucz publiczny (asymetryczny) Kryptografia

Machine Translated by Google

Rozdział 8

Teoria liczb i kryptografia Założenia dotyczące twardości

Współczesne kryptosystemy niezmiennie opierają się na założeniu, że jakiś problem jest trudny. Na przykład w rozdziałach 3 i 4 widzieliśmy, że kryptografia klucza prywatnego – zarówno schematy szyfrowania, jak i kody uwierzytelniania wiadomości – może opierać się na założeniu, że istnieją permutacje pseudolosowe (inaczej szyfry blokowe). Przypomnijmy, w przybliżeniu oznacza to, że istnieje pewna permutacja F z kluczem, dla której trudno jest rozróżnić w czasie wielomianowym interakcje z F_k (dla jednolitego, nieznanego klucza k) od interakcji z permutacją prawdziwie losową.

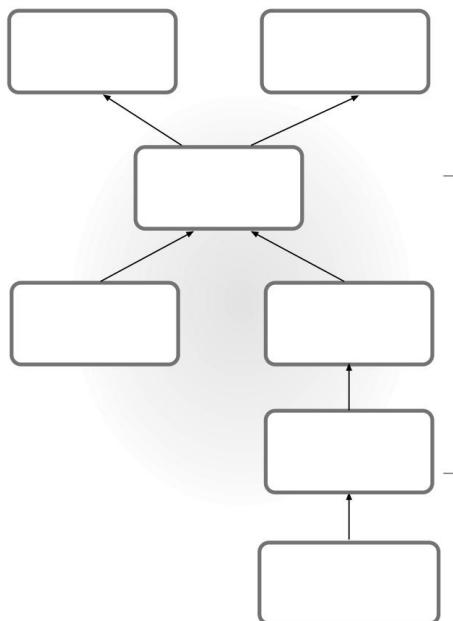
Na pierwszy rzut oka założenie o istnieniu permutacji pseudolosowych wydaje się dość mocne i nienaturalne, w związku z czym zasadne jest pytanie, czy założenie to jest prawdziwe i czy istnieją dowody na jego poparcie. W rozdziale 6 zbadaliśmy, jak w praktyce zbudowane są szyfry blokowe. Fakt, że istniejące konstrukcje są odporne na atak, służy jako wskazówka, że istnienie permutacji pseudolosowych jest prawdopodobne. Mimo to może trudno uwierzyć, że nie ma skutecznych ataków rozróżniających na istniejące szyfry blokowe. Co więcej, obecny stan naszej teorii jest taki, że nie wiemy, jak udowodnić pseudolosowość którykolwiek z istniejących konstrukcji praktycznych w porównaniu z jakimkolwiek „prostszym” lub „rozsądniejszym” założeniem. W sumie nie jest to stan do końca satysfakcyjny.

Dla kontrastu, jak wspomniano w Rozdziale 3 (i szczegółowo zbadaliśmy w Rozdziale 7), możliwe jest udowodnienie istnienia permutacji pseudolosowych w oparciu o znacznie łagodniejsze założenie, że istnieją funkcje jednokierunkowe. (Nieformalnie funkcja jest jednokierunkowa, jeśli można ją łatwo obliczyć, ale trudno odwrócić; patrz podrozdział 8.4.1.) Jednakże poza krótkim omówieniem w podrozdziale 7.1.2 nie widzieliśmy konkretnych przykładów funkcji uważa się, że jest jednokierunkowy.

Jednym z celów tego rozdziału jest przedstawienie różnych problemów uznawanych za „trudne” i zaprezentowanie przypuszczalnych funkcji jednokierunkowych opartych na tych problemach.¹ Ten rozdział można zatem postrzegać jako kulminację „odgórnego” podejścia do problemu kryptografia klucza prywatnego. (Patrz rysunek 8.1.) Oznacza to, że w rozdziałach 3 i 4 pokazaliśmy, że kryptografia klucza prywatnego może opierać się na funkcjach i permutacjach pseudolosowych. Widzieliśmy wówczas, że to drugie

¹Przypomnijmy, że obecnie nie wiemy, jak udowodnić istnienie funkcji jednokierunkowych, więc najlepsze, co możemy zrobić, to oprzeć funkcje jednokierunkowe na założeniach dotyczących trudności niektórych problemów.

można utworzyć w praktyce przy użyciu szyfrów blokowych, jak omówiono w rozdziale 6, lub można je skonstruować w rygorystyczny sposób z dowolnej funkcji jednokierunkowej, jak pokazano w rozdziale 7. Tutaj pójdziemy o krok dalej i pokażemy, jak funkcje jednokierunkowe można oprzeć na pewnych trudnych problemach matematycznych.



RYSUNEK 8.1: Kryptografia klucza prywatnego: podejście odgórne.

Przykłady, które badamy, mają charakter teorii liczb i dlatego też rozpoczęń od krótkiego wprowadzenia do teorii liczb i teorii grup. Ponieważ interesują nas także problemy, które można sprawnie rozwiązać (nawet np funkcja jednokierunkowa musi być łatwa do obliczenia w jednym kierunku, a schemat kryptograficzny musi dopuszczać wydajne algorytmy dla uczciwych stron), zainicjować także badania nad algorytmiczną teorią liczb. Nawet czytelnik, który nim jest Zaznajomiony z teorią liczb lub teorią grup, zachęca się do przeczytania tego rozdziału, ponieważ aspekty algorytmiczne są zazwyczaj ignorowane w czysto matematycznym podejściu traktowanie tych tematów.

Drugim celem tego rozdziału jest opracowanie materiału potrzebnego do klucza publicznego kryptografii, której badanie zaczniemy w rozdziale 10. Choć jest to uderzające w ustawnieniu klucza prywatnego istnieją wydajne konstrukcje tego, co niezbędne prymitywy (szyfry blokowe i funkcje skrótu) bez wywoływania jakiegokolwiek liczby teorii, w ustawnieniu klucza publicznego wszystkie znane konstrukcje opierają się na trudnych problemach z teorią liczb. Materiał zawarty w tym rozdziale służy zatem zarówno jako kulminacja tego, czego do tej pory badaliśmy w odniesieniu do kryptografii klucza prywatnego, jak i a także podstawa kryptografii klucza publicznego.

8.1 Wstęp i podstawowa teoria grup

Zaczynamy od przeglądu liczb pierwszych i podstawowej arytmetyki modułowej.

Nawet czytelnik, który widział już te tematy, powinien przejrzeć kolejne dwie sekcje, ponieważ część materiału może być nowa i załączamy dowody na większość podanych wyników.

8.1.1 Liczby pierwsze i podzielność

Zbiór liczb całkowitych jest oznaczony przez \mathbb{Z} . Dla $a, b \in \mathbb{Z}$ mówimy, że a dzieli b , zapisując $a | b$, jeśli istnieje liczba całkowita c taka, że $ac = b$. Jeżeli a nie dzieli b , piszemy $a \nmid b$. (Interesuje nas przede wszystkim przypadek, w którym wszystkie a, b i c są dodatnie, chociaż definicja ma sens nawet wtedy, gdy jedna lub więcej z nich jest ujemna lub równa zero.) Prostą obserwacją jest to, że jeśli $a | b$ i $a | c$ następnie $a | (xb + yc)$ dla dowolnego $x, y \in \mathbb{Z}$.

Jeśli $|b| > 1$ i a jest dodatnie, nazywamy a dzielnikiem b . Jeśli dodatkowo $a \in \{1, b\}$, to a nazywa się nietrywialnym dzielnikiem lub współczynnikiem b . Dodatnia liczba całkowita $p > 1$ jest liczbą pierwszą, jeśli nie ma czynników; tj. ma tylko dwa dzielniaki: 1 i siebie. Dodatnia liczba całkowita większa od 1, która nie jest liczbą pierwszą, nazywana jest złożoną. Zgodnie z konwencją liczba 1 nie jest ani liczbą pierwszą, ani złożoną.

Podstawowym twierdzeniem arytmetyki jest to, że każdą liczbę całkowitą większą niż 1 można wyrazić jednoznacznie (aż do uporządkowania) jako iloczyn liczb pierwszych. To dowolna dodatnia liczba całkowita $N > 1$ może być zapisana jako $N = p_1^{e_1} \cdots p_n^{e_n}$, gdzie $\{p_i\}$ to różne liczbami pierwszymi i $e_i \geq 1$ dla wszystkich i ; ponadto $\{p_i\}$ ($i \in \{1, \dots, n\}$) są jednoznacznie określone aż do zamówienia.

Proces dzielenia reszty z elementu elementarnego jest nam znany z szkoły. Poniższe twierdzenie formalizuje to pojęcie.

TWIERDZENIE 8.1 Niech a będzie liczbą całkowitą, a b będzie liczbą całkowitą dodatnią. Istnieją wówczas unikalne liczby całkowite q, r dla których $a = qb + r$ oraz $0 \leq r < b$.

Ponadto, mając liczby całkowite a i b , jak w twierdzeniu, można obliczyć q i r w czasie wielomianowym; patrz dodatek B.1. (Czas działania algorytmu jest mierzony jako funkcja długości jego danych wejściowych. Ważnym punktem w kontekście algorytmicznej teorii liczb jest to, że zawsze zakłada się, że dane wejściowe w postaci liczb całkowitych są reprezentowane w postaci binarnej. Czas działania algorytmu algorytm przyjmujący jako dane wejściowe liczbę całkowitą N jest zatem mierzony w kategoriach N , czyli długości binarnej reprezentacji N . Należy zauważać, że $N = \log N + 1$.)

Największy wspólny dzielnik dwóch liczb całkowitych a, b , zapisany $\gcd(a, b)$, jest największą liczbą całkowitą c taką, że $c | a$ i $c | b$. (Pozostawiamy $\gcd(0, 0)$ niezdefiniowanym.) Pojęcie największego wspólnego dzielnika ma sens, gdy jedno lub oba z a, b są ujemne, ale zazwyczaj będziemy mieli $a, b \in \mathbb{N}$; w każdym razie $\gcd(a, b) = \gcd(|a|, |b|)$.

Zauważ, że $\gcd(b, 0) = \gcd(0, b) = b$; także, jeśli p jest liczbą pierwszą, to $\gcd(a, p)$ jest albo równe 1, albo p . Jeśli $\gcd(a, b) = 1$, mówimy, że a i b są względnie pierwsze.

Poniżej znajduje się przydatny wynik:

TWIERDZENIE 8.2 Niech a, b będą dodatnimi liczbami całkowitymi. Istnieją wówczas liczby całkowite X, Y takie, że $Xa + Yb = \gcd(a, b)$. Ponadto $\gcd(a, b)$ jest najmniejszą dodatnią liczbą całkowitą, którą można wyrazić w ten sposób.

DOWÓD Rozważmy zbiór $I = \{Xa^+ + Yb^+ \mid Z\}$. Zauważ, że $a, b \in I$, więc I z pewnością zawiera pewne dodatnie liczby całkowite. Niech d będzie najmniejszą dodatnią liczbą całkowitą w I . Pokazujemy, że $d = \gcd(a, b)$; ponieważ d można zapisać jako $d = Xa + Yb$ dla pewnego $X, Y \in I$ (ponieważ $d \in I$), to dowodzi twierdzenia.

Aby to pokazać, musimy udowodnić, że $d \mid a$ i $d \mid b$ i że d jest największą liczbą całkowitą o tej własności. W rzeczywistości możemy pokazać, że d dzieli każdy element w I . Aby to zobaczyć, weź dowolne $c \in I$ i napisz $c = Xa + Yb$ z $Y \in I$. Korzystając z dzielenia z resztą (Twierdzenie 8.1)

, mamy, że X

$c = qd + r$ z liczbami całkowitymi $q, r \in I$ i $0 \leq r < d$. Następnie

$$r = do - qd = Xa + Yb - q(Xa + Yb) = (X - qX)a + (Y - qY)b \in I.$$

Jeśli $r = 0$, jest to sprzeczne z naszym wyborem d jako najmniejszej dodatniej liczby całkowitej w I (ponieważ $r < d$). Zatem $r = 0$, a zatem $d \mid c$. To pokazuje, że d dzieli każdy element I .

Ponieważ $a \in I$ i $b \in I$, z powyższego wynika, że $d \mid a$ i $d \mid b$ i tak d jest wspólnym dzielnikiem a i b . Pozostaje pokazać, że jest to największy wspólny dzielnik. Założmy, że istnieje liczba całkowita $d' > d$ taka, że $d' \mid a$ i $d' \mid b$. Następnie, zgodnie z wcześniejszą obserwacją, $d' \mid Xa + Yb$. Ponieważ ta ostatnia jest równa d , oznacza to $d' \mid d$. Ale jest to niemożliwe, jeśli d jest większe niż d' . Dochodzimy do wniosku, że d jest największą liczbą całkowitą dzielącą zarówno a , jak i b , a zatem $d = \gcd(a, b)$. ■

Biorąc pod uwagę a i b , algorytm Euklidesa może zostać użyty do obliczenia $\gcd(a, b)$ w czasie wielomianowym. Rozszerzony algorytm Euklidesa można zastosować do obliczenia X, Y (jak w powyższym twierdzeniu) również w czasie wielomianowym. Szczegóły podano w Załączniku B.1.2.

Powyższe twierdzenie jest bardzo przydatne w udowadnianiu dodatkowych wyników dotyczących podzielności. Pokażemy teraz dwa przykłady.

TWIERDZENIE 8.3 Jeśli $c \mid ab$ i $\gcd(a, c) = 1$, następnie $c \mid b$. Zatem, jeśli p jest liczbą pierwszą i $p \mid ab$, a następnie albo $p \mid a$ lub $p \mid b$.

DOWÓD Od ok. $|ab$ mamy $yc = ab$ dla pewnej liczby całkowitej y . Jeśli $\gcd(a, c) = 1$, to z poprzedniego twierdzenia wiemy, że istnieją liczby całkowite X, Y takie, że

$1 = Xa + Y \cdot b$. Mnożąc obie strony przez b , otrzymujemy

$$b = Xab + Y \cdot cb = Xyc + Y \cdot cb = c \cdot (Xy + Yb).$$

Ponieważ $(Xy + Yb)$ jest liczbą całkowitą, wynika z tego, że $c \mid b$.

Druga część twierdzenia wynika z faktu, że jeśli $p \mid a$ następnie $\gcd(a, p) = 1$.

TWIERDZENIE 8.4 Jeżeli $|N, b | N$ i $\gcd(a, b) = 1$, następnie $ab | N$.

DOWÓD Zapisz $ac = N, bd = N$ i (korzystając z Twierdzenia 8.2) $1 = Xa + Yb$, gdzie c, d, X, Y są liczbami całkowitymi. Mnożąc obie strony ostatniego równania przez N , otrzymujemy

$$N = XaN + YbN = Xabd + Ybac = ab(Xd + Yc),$$

pokazując, że $ab | N$.

8.1.2 Arytmetyka modułowa

Niech $a, b, N \in \mathbb{Z}$ z $N > 1$. Zapisu $[a \bmod N]$ używamy do oznaczenia reszty a z dzielenia przez N . Bardziej szczegółowo: zgodnie z Twierdzeniem 8.1 istnieje jednoznaczne q, r z $a = qN + r$ i $0 \leq r < N$ i definiujemy $[a \bmod N]$ jako równe temu r . Należy zatem zauważyć, że $0 \leq [a \bmod N] < N$. Proces mapowania a na $[a \bmod N]$ nazywamy modułem redukcji N .

Mówimy, że a i b są przystające modulo N , zapisane $a = b \bmod N$, jeśli $[a \bmod N] = [b \bmod N]$, tj. jeśli reszta z dzielenia a przez N jest taka sama jak reszta z b jest dzielony przez N . Zauważ, że $a = b \bmod N$ wtedy i tylko wtedy, gdy $N \mid (a - b)$. Notacyjnie, w wyrażeniu takim jak

$$a = b = c = \dots = z \bmod N,$$

należy rozumieć, że każdy znak równości w tej sekwencji (a nie tylko ostatni) odnosi się do kongruencji modulo N .

Zauważ, że $a = [b \bmod N]$ implikuje $a = b \bmod N$, ale nie odwrotnie. Na przykład $36 = 21 \bmod 15$, ale $36 = [21 \bmod 15] = 6$. (Z drugiej strony $[a \bmod N] = [b \bmod N]$ wtedy i tylko wtedy, gdy $a = b \bmod N$.)

Kongruencja modulo N jest relacją równoważności: tj. jest zwrotna ($a = a \bmod N$ dla wszystkich a), symetryczna ($a = b \bmod N$ implikuje $b = a \bmod N$) i przechodnia (jeśli $a = b \bmod N$ i $b = c \bmod N$, następnie $a = c \bmod N$). Kongruencja modulo N podlega również standardowym zasadom arytmetyki w odniesieniu do dodawania, odejmowania i mnożenia; więc na przykład, jeśli $a = a \bmod N$ i $b = b \bmod N$, to $(a + b) = (a + b) \bmod N$ i $ab = ab \bmod N$. Konsekwencją jest to, że możemy „zredukować”, a następnie dodawać/mnożyć” zamiast „dodawać/mnożyć, a następnie zmniejszać”, co często może uprościć obliczenia.

Przykład 8.5

Obliczmy $[1093028 \cdot 190301 \bmod 100]$. Ponieważ $1093028 = 28 \bmod 100$ i $190301 = 1 \bmod 100$, mamy

$$\begin{aligned}1093028 \cdot 190301 &= [1093028 \bmod 100] \cdot [190301 \bmod 100] \bmod 100 \\&= 28 \cdot 1 = 28 \bmod 100.\end{aligned}$$

Alternatywny sposób obliczenia odpowiedzi (tj. obliczenie iloczynu $1093028 \cdot 190301$, a następnie zmniejszenie wyniku modulo 100) jest mniej efektywny.

Kongruencja modulo N (ogólnie) nie uwzględnia podziału. Oznacza to, że jeśli $a = a \bmod N$ i $b = b \bmod N$, to niekoniecznie jest prawdą, że $a/b = a/b \bmod N$; w rzeczywistości wyrażenie „ $a/b \bmod N$ ” nie zawsze jest dobrze zdefiniowane.

Jako konkretny przykład, który często powoduje zamieszanie, $ab = cb \bmod N$ niekoniecznie oznacza, że $a = c \bmod N$.

Przykład 8.6

Weźmy $N = 24$. Następnie $3 \cdot 2 = 6 = 15 \cdot 2 \bmod 24$, ale $3 = 15 \bmod 24$.

Jednakże w niektórych przypadkach możemy zdefiniować znaczące pojęcie podziału.

Jeśli dla danej liczby całkowitej b istnieje liczba całkowita c taka, że $bc = 1 \bmod N$, mówimy, że b jest odwracalne modulo N i nazywamy c (multiplikatywną) odwrotnością b modulo N . Jasne jest, że 0 nigdy nie jest odwracalne. Nie jest trudno pokazać, że jeśli c jest multiplikatywną odwrotnością b modulo N , to taka jest również $[c \bmod N]$. Ponadto, jeśli c jest kolejną multiplikatywną odwrotnością b , to $[c \bmod N] = [c \bmod N]$. Kiedy b jest odwracalne, możemy po prostu pozwolić, aby b oznaczało unikalną multiplikatywną odwrotność \bar{b} , która leży w zakresie $\{1, \dots, N - 1\}$.

Gdy b jest odwracalne modulo N , definiujemy dzielenie przez b modulo N jako $\text{mul-} = [ab - 1] \bmod N$ (dzielenie przez b $\overset{1}{[a/b \bmod N]}$ typowanie przez b , że $\overset{\text{def}}{b^{-1} \bmod N}$). Podkreślamy (tj. definiujemy) jest zdefiniowane tylko wtedy, gdy b jest odwracalne. Jeśli $ab = cb \bmod N$ i b jest odwracalne, wówczas możemy podzielić każdą stronę równania przez b (lub, tak naprawdę, pomnóż każdą stronę przez b^{-1}) pozyskać

$$(ab) \cdot b^{-1} = (cb) \cdot b^{-1} \bmod N \quad a = ab \bmod N.$$

Widzimy, że w tym przypadku dzielenie działa „zgodnie z oczekiwaniami”. Dlatego w pewnym sensie „łatwiej” jest pracować z odwracalnymi liczbami całkowitymi modulo N .

Naturalnym pytaniem jest: które liczby całkowite są odwracalne modulo przy danej modułus N ? Możemy w pełni odpowiedzieć na to pytanie, korzystając z Twierdzenia 8.2:

TWIERDZENIE 8.7 Niech b, N będą liczbami całkowitymi, gdzie $b \neq 1$ i $N > 1$. Wtedy b jest odwracalne modulo N wtedy i tylko wtedy, gdy $\gcd(b, N) = 1$.

DOWÓD Założymy, że b jest odwracalne modulo N i niech c oznacza jego odwrotność.

Ponieważ $bc = 1 \bmod N$, oznacza to, że $bc - 1 = yN$ dla pewnego y . Z. Równoważ.-

dokładnie, $b \cdot c \equiv N = 1$. Ponieważ zgodnie z Twierdzeniem 8.2 $\gcd(b, N)$ jest najmniejszą dodatnią liczbą całkowitą, którą można wyrazić w ten sposób, i nie ma dodatniej liczby całkowej mniejszej niż 1, oznacza to, że $\gcd(b, N) = 1$.

I odwrotnie, jeśli $\gcd(b, N) = 1$, to zgodnie z Twierdzeniem 8.2 istnieją liczby całkowite X, Y takie, że $Xb + YN = 1$. Redukcja każdej strony tego równania modulo N daje $Xb = 1 \pmod{N}$ i widzimy, że X jest multiplikatywna odwrotność b . (W rzeczywistości daje to wydajny algorytm do obliczania odwrotności.) ■

Przykład 8.8 Niech

$b = 11$ i $N = 17$. Wtedy $(-3) \cdot 11 + 2 \cdot 17 = 1$, a więc $14 = [-3 \pmod{17}]$ jest odwrotnością 11. Można sprawdzić, że $14 \cdot 11 = 1 \pmod{17}$.

Dodawanie, odejmowanie, mnożenie i obliczanie odwrotności (jeśli istnieją) modulo N można przeprowadzić w czasie wielomianowym; patrz dodatek B.2. Potęgowanie (tzn. obliczanie $[a^b] \pmod{N}$ dla $b > 0$ liczba całkowita) Dodatek B.2.3.

8.1.3 Grupy

Niech G będzie zbiorem. Operacja binarna \cdot na G jest po prostu funkcją (\cdot, \cdot) , która przyjmuje na wejściu dwa elementy G . Jeśli $g, h \in G$ to zamiast używać kłopotliwej notacji (g, h) piszemy $g \cdot h$.

Wprowadzimy teraz ważne pojęcie grupy.

DEFINICJA 8.9 Grupa to zbiór G wraz z operacją binarną \cdot , dla którego spełnione są następujące warunki:

- (Zamknięcie:) Dla wszystkich $g, h \in G$, $g \cdot h \in G$.
- (Istnienie tożsamości :) Istnieje tożsamość $e \in G$ taka, że dla wszystkich $g \in G$, $g \cdot e = e \cdot g = g$.
- (Istnienie odwrotności :) Dla każdego $g \in G$ istnieje element $h \in G$ taki, że $g \cdot h = h \cdot g = e$.
- (Skajarzenie:) Dla wszystkich $g_1, g_2, g_3 \in G$, $(g_1 \cdot g_2) \cdot g_3 = g_1 \cdot (g_2 \cdot g_3)$.

Gdy G ma skończoną liczbę elementów, mówimy, że G jest skończone i niech $|G|$ oznaczają rząd grupy (to znaczy liczbę elementów w G).

Grupa G z działaniem \cdot jest abelowa, jeśli zachodzi:

- (Przemienność:) Dla wszystkich $g, h \in G$, $g \cdot h = h \cdot g$.

Kiedy zrozumiemy operację binarną, po prostu nazywamy zbiór G grupą.

Zawsze będziemy mieć do czynienia ze skończonymi grupami abelowymi. Będziemy ostrożnie określać, jednakże, gdy wynik wymaga tych założeń.

Łączność oznacza, że nie musimy dołączać nawiasów podczas pisania długich wyrażeń; czyli zapis $g_1 \ g_2 \ \dots \ g_n$ jest jednoznaczny, ponieważ nie ma znaczenia, w jakiej kolejności oceniamy operację .

Można wykazać, że element tożsamości w grupie G jest unikalny i dlatego możemy odnosić się do tożsamości grupy. Można również pokazać, że każdy element g grupy ma unikalną odwrotność. Zobacz ćwiczenie 8.1.

Jeśli G jest grupą, zbiór H $\subseteq G$ jest podgrupą G, jeśli H sam tworzy grupę w wyniku tej samej operacji związanej z G. Aby sprawdzić, czy H jest podgrupą, musimy sprawdzić domknięcie, istnienie tożsamości i odwrotności, i łączność zgodnie z definicją 8.9. (Wrzeczywistości łączność – jak również przemienność, jeśli G jest abelowa – jest dziedziczona automatycznie z G.) Każda grupa G zawsze ma trywialne podgrupy G i $\{1\}$. Nazywamy H ścisłą podgrupą G, jeśli $H = G$.

Ogólnie rzecz biorąc, nie będziemy używać notacji \cdot do oznaczenia działania grupowego. Zamiast tego użyjemy notacji addytywnej lub notacji mnożonej, w zależności od omawianej grupy. Nie oznacza to, że operacja grupowa odpowiada dodawaniu lub mnożeniu liczb całkowitych; jest to jedynie użyteczna notacja. W przypadku stosowania notacji addytywnej operacja grupowa zastosowana do dwóch elementów g, h jest oznaczona jako $g + h$; tożsamość jest oznaczona przez 0; odwrotność elementu g jest oznaczona przez $-g$; i piszemy $h - g$ zamiast $h + (-g)$. W przypadku stosowania notacji mnożonej operacja grupowa zastosowana do g, h jest oznaczona przez $g \cdot h$ lub po prostu gh ; tożsamość jest oznaczona przez 1; odwrotność elementu g jest oznaczona przez g^{-1} ; i czasami piszemy h/g zamiast hg^{-1} .

W tym momencie pomocne może być zapoznanie się z kilkoma przykładami.

Przykład 8.10 Zbiór

może być grupą podlegającą jednej operacji, ale nie inną. Na przykład zbiór liczb całkowitych Z jest dodawaną grupą abelową: tożsamością jest element 0, a każda liczba całkowita g ma odwrotność $-g$. Z drugiej strony nie jest to grupa podlegająca mnożeniu, ponieważ na przykład liczba całkowita 2 nie ma odwrotności mnożonej w liczbach całkowitych.

Przykład 8.11

Zbiór liczb rzeczywistych R nie jest grupą podlegającą mnożeniu, ponieważ 0 nie ma odwrotności mnożonej. Jednakże zbiór niezerowych liczb rzeczywistych jest grupą abelową podlegającą mnożeniu przez tożsamość 1.

Poniższy przykład przedstawia grupę Z_N , z której będziemy często korzystać.

Przykład 8.12

Niech $N > 1$ będzie liczbą całkowitą. Zbiór $\{0, \dots, N-1\}$ w odniesieniu do dodawania modulo N (tj. gdzie $a+b = [a+b \text{ mod } N]$) jest grupą abelową rzędu N . Zamknięcie jest oczywiste; łączność i przemienność wynikają z faktu, że liczby całkowite spełniają te własności; tożsamość wynosi 0; i ponieważ $a+ (N-a) = 0 \text{ mod } N$, wynika z tego, że odwrotnością dowolnego elementu a jest $[(N-a) \text{ mod } N]$. My

oznacz tę grupę przez ZN . (Czasami będziemy także używać ZN do oznaczenia zbioru $\{0, \dots, N-1\}$ bez względu na jakąkolwiek konkretną operację na grupie.)

Kończymy tę sekcję łatwym lematem, który formalizuje „prawo anulowania” dla grup.

LEMMA 8.13 Niech G będzie grupą oraz $a, b, c \in G$. Jeśli $ac = bc$, to $a = b$.

W szczególności, jeśli $ac = c$, to a jest tożsamością w G .

DOWÓD Wiemy, że $ac = bc$. Mnożąc obie strony przez unikalną odwrotność c , otrzymujemy $a = b$.
 w⁻¹ Szczegółowo:

$$ac = bc \quad (ac)c \quad ^1 = (bc) \cdot do \quad ^1 \quad a(cc-1) = b(cc-1) \quad a \cdot 1 = b \cdot 1$$

a = b

1

Porównaj powyższy dowód z dyskusją (poprzedzającą Twierdzenie 8.7) dotyczącą prawa anulowania dzielenia modulo N. Jak wskazuje podobieństwo, elementy odwzoracalne modulo N tworzą grupę pod mnożeniem modulo N. Powrócimy do tego przykładu wkrótce bardziej szczegółowo.

Potęgowanie grupowe

Często przydatna jest możliwość opisania operacji grupowej zastosowanej m razy do stałego elementu g , gdzie m jest dodatnią liczbą całkowitą. Używając zapisu addytywnego, wyrażamy to jako $m \cdot g$ lub mg ; to jest,

$$mg = m \cdot g \stackrel{\text{def}}{=} \underline{\text{sol}} + \dots + \underline{\text{sol}} \quad \text{m razy}$$

Zauważ, że m jest liczbą całkowitą, a g jest elementem grupy. Zatem mg nie reprezentuje operacji grupowej zastosowanej do m i g (w rzeczywistości pracujemy w grupie, w której operacja grupowa jest zapisywana addytywnie). Na szczęście jednak zapis „zachowuje się tak, jak powinien”; więc na przykład, jeśli $g \in G$ i m, m są liczbami całkowitymi, to $(mg) + (mg) = (m + m)g$, $m(mg) = (mm)g$ i $1 \cdot g = g$. W grupie abelowej G z $g, h \in G$, $(mg) + (mh) = m(g + h)$.

Używając notacji mnożenia skróconego, wyrażamy zastosowanie operacji grupowej m razy do elementu q przez q m . To jest,

$$G^M \stackrel{\text{def}}{=} g \cdots g$$

— —
m razy

Znane zasady potegowania obowiązują: $a^m \cdot a^m = a^{m+m}$ i $(a^m)^n = a^{m \cdot n}$

¹ $\equiv g$. Ponadto, jeśli G jest grupą abelową i $a, b \in G$, to $a \cdot m \cdot b = (ab) \cdot m$.

Wszystko to są po prostu „tłumaczenia” wyników z poprzedniego akapitu na ustalenie grup zapisanych mnożnikowo, a nie addytywnie.

Powyższy zapis zostaje w naturalny sposób rozszerzony na przypadek, gdy m wynosi zero lub ujemna liczba całkowita. Używając notacji addytywnej, definiujemy $0 \cdot g = (-m) \cdot g$ dla małej liczby całkowitej. (Zauważ, że w równaniu po lewej stronie jest liczbą całkowitą 0, podczas gdy 0 po prawej stronie jest elementem tożsamości.) Należy zauważyć, że $-g$ jest odwrotnością g , jak można się spodziewać, $(-m) \cdot g = (mg)$. Używając mnożnikowego m . Ponownie, G

$$\text{def } , g = 1 \text{ i } g \stackrel{0}{\text{mały}} \text{ g Niech } \stackrel{m}{=} (\text{np. } 1) \text{ jest odwrotnością } g \text{ i my } g \stackrel{G}{\in} \stackrel{m}{=} (gm).$$

b) 0 będą liczbą całkowitą. Następnie potęgowanie g można obliczyć za pomocą liczby wielomianowej podstawowych operacji grupowych w G .

Zatem, jeśli operację grupową można obliczyć w czasie wielomianowym, to możliwe jest także potęgowanie. Omówiono to w dodatku B.2.3.

Wiemy już wystarczająco dużo, aby udowodnić następujący niezwykły wynik:

TWIERDZENIE 8.14 Niech G będzie grupą skończoną, gdzie $m = |G|$, rzad grupy. Następnie dla dowolnego elementu $g \in G$, $g^m = 1$.

DOWÓD Twierdzenie dowodzimy tylko wtedy, gdy G jest abelowe (choć zachodzi dla dowolnej grupy skończonej). Ustal dowolne $g \in G$ i niech g_1, \dots, g_m będą elementami G . Twierdzimy, że

$$g_1 \cdot g_2 \cdots g_m = (gg_1) \cdot (gg_2) \cdots (gg_m).$$

Aby to zobaczyć, zauważ, że $ggi = ggj$ implikuje $gi = gj$ zgodnie z Lematem 8.13. Zatem każdy z m elementów w nawiasach po prawej stronie jest odrębny. Ponieważ w G jest dokładnie m elementów, m elementów mnożonych przez siebie po prawej stronie to po prostu wszystkie elementy G w jakiejś permutowanej kolejności. Ponieważ G jest abelowe, kolejność mnożenia elementów nie ma znaczenia, więc prawa strona jest równa lewej stronie.

Ponownie wykorzystując fakt, że G jest abelowe, możemy „wyciągnąć” wszystkie wystąpienia g i otrzymać

$$g_1 \cdot g_2 \cdots g_m = (gg_1) \cdot (gg_2) \cdots (gg_m) = g^m \cdot (g_1 \cdot g_2 \cdots g_m).$$

Odwołując się ponownie do Lematu 8.13, oznacza to, że $g^m = 1$. ■

Ważnym następstwem powyższego jest to, że możemy pracować „modulo” z grupą porządek” w wykładniku:

WNIOSEK 8.15 Niech G będzie grupą skończoną, w której $m = |G| > 1$. Wtedy dla dowolnego $g \in G$ i dowolnej liczby całkowitej x mamy $g^x = g^{[x \bmod m]}$.

DOWÓD Powiedzmy, że $x = qm+r$, gdzie q, r są liczbami całkowitymi, a $r = [x \bmod m]$. Następnie

$$G^x = g^{qm+r} \stackrel{m}{=} (g^m)^q \cdot g^r = 1 \cdot g^r = g^r$$

(używając Twierdzenia 8.14), jak twierdzono. ■

Przykład 8.16

Zapisany addytywnie powyższy wniosek mówi, że jeśli g jest elementem grupy rzędu m , to $x \cdot g = [x \bmod m] \cdot g$. Jako przykład rozważmy grupę Z_{15} rzędu $m = 15$ i przyjmijmy $g = 11$. Wniosek mówi, że

$$152 \cdot 11 = [152 \bmod 15] \cdot 11 = 2 \cdot 11 = 11 + 11 = 22 = 7 \bmod 15.$$

Powyższe zgadza się z faktem (por. przykład 8.5), że możemy „zmniejszać, a następnie pomnażać”, zamiast „mnożyć, a następnie zmniejszać”.

Kolejnym wnioskiem, który będzie niezwykle przydatny w zastosowaniach kryptograficznych, jest następujący:

WNIOSZEK 8.17 Niech G będzie grupą skończoną, w której $m = |G| > 1$. Niech $e > 0$ będzie liczbą całkowitą i zdefiniuje funkcję $f : G \rightarrow G$ przez $f(e)(g) = g^e$. Jeżeli $\gcd(e, m) = 1$, to f jest permutacją (tzn. bijekcją). Co więcej, jeśli $d = e \bmod m$, to f_d jest odwrotnością f_e . (Uwaga do Twierdzenia 8.7, $\gcd(e, m) = 1$ oznacza, że e jest odwracalne modulo m .)

DOWÓD Ponieważ G jest skończona, druga część twierdzenia implikuje pierwszą; zatem musimy jedynie pokazać, że f_d jest odwrotnością f_e . Jest to prawda, ponieważ dla dowolnego $g \in G$ mamy

$$f_d(f_e(g)) = f_d(g^e) = (g^e)^d \stackrel{re \ ed \ mod \ m}{=} g^{ed} = g \quad 1 = g^{\frac{m}{\gcd(e,m)}} = g,$$

gdzie czwarta równość wynika z Wniosku 8.15. ■

8.1.4 Grupa Z_N

Jak omówiono w przykładzie 8.12, zbiór $Z_N = \{0, \dots, N-1\}$ jest grupą podlegającą dodaniu modulo N . Czy możemy zdefiniować grupę ze względu na mnożenie modulo N ? Robiąc to, będziemy musieli wyeliminować te elementy w Z_N , które nie są odwracalne; np. będącym musieli wyeliminować 0, ponieważ nie ma ono odwrotności mnożonej. Elementy niezerowe mogą również nie być odwracalne (por. Twierdzenie 8.7).

Które elementy $b \in \{1, \dots, N-1\}$ są odwracalne modulo N ? Twierdzenie 8.7 mówi, że są to dokładnie te elementy b , dla których $\gcd(b, N) = 1$. Mamy

widać także w Sekcji 8.1.2, że ilekroć b jest odwracalne, ma ono odwrotność mieszącą się w przedziale $\{1, \dots, N - 1\}$. Prowadzi nas to do zdefiniowania zbioru dla dowolnego $N > 1$

$$\mathbb{Z}_N \stackrel{\text{def}}{=} \{b \in \{1, \dots, N - 1\} \mid \gcd(b, N) = 1\};$$

tj. \mathbb{Z}_N składa się z liczb całkowitych ze zbioru $\{1, \dots, N - 1\}$, które są względnie pierwsze do N . Operacją grupową jest mnożenie modulo N ; tj. ab Twierdzimy, że $\mathbb{Z} \stackrel{\text{def}}{=} [ab \bmod N]$.

\mathbb{Z}_N jest grupą abelową w odniesieniu do tej operacji. zestaw Ponieważ 1 jest zawsze w \mathbb{Z}_N , wyraźnie zawiera element tożsamości. The ma multiplikatywną \mathbb{Z}_N , powyższa dyskusja pokazuje, że każdy element w \mathbb{Z}_N odwrotność w tym samym zestawie. Przemienność i łączność wynikają z faktu, że te właściwości obowiązują w przypadku liczb całkowitych. Aby pokazać, że zachodzi domknięcie, niech N ; wtedy $[ab \bmod a, b] \subset \mathbb{Z}_N$ ma odwrotność $[b^{-1}a^{-1} \bmod N]$, co oznacza, że N . Zreasumowanie: $\gcd([ab \bmod N], N) = 1$ i tak $ab \in \mathbb{Z}_N$

TWIERDZENIE 8.18 Niech $N > 1$ będzie liczbą całkowitą. Następnie grupa \mathbb{Z}_N jest abelem \mathbb{Z} pod mnożeniem modulo N .

Zdefiniuj $\varphi(N) \stackrel{\text{def}}{=} |\mathbb{Z}_N|$, rzad grupy \mathbb{Z}_N . (φ nazywa się funkcją Eulera phi.) Jaka jest wartość $\varphi(N)$? Rozważmy najpierw przypadek, gdy $N = p$ jest liczbą pierwszą. Następnie wszystkie elementy w $\{1, \dots, p - 1\}$ są względnie pierwsze do p , więc $\varphi(p) = |\mathbb{Z}_p| = p - 1$. Następnie rozważmy przypadek, że $N = pq$, gdzie p, q są różnymi liczbami pierwszymi. Jeśli liczba całkowita $a \in \{1, \dots, N - 1\}$ nie jest względnie pierwszą liczbą N , to albo $p \mid a$ lub $q \mid a$ (a nie może być podzielne przez p i q , ponieważ oznaczałoby to $pq \mid a$, ale $a < N = pq$). Elementy w $\{1, \dots, N - 1\}$ podzielne przez p to dokładnie $(p - 1)$ elementy $p, 2p, 3p, \dots, (q - 1)p$, a elementy podzielne przez q to dokładnie $(p - 1)$ elementy $q, 2q, \dots, (p - 1)q$. Liczba pozostałych elementów (tj. tych, które nie są podzielne przez p ani q) jest zatem dana przez

$$(N - 1) - (q - 1) - (p - 1) = pq - p - q + 1 = (p - 1)(q - 1).$$

W ten sposób udowodniliśmy, że $\varphi(N) = (p - 1)(q - 1)$, gdy N jest iloczynem dwóch różnych liczb pierwszych p i q .

Zostaniesz poproszony o udowodnienie następującego ogólnego wyniku (używanego rzadko w reszta książki) w Ćwiczeniu 8.4:

TWIERDZENIE 8.19 Niech $N = e_1^{e_1} \cdots e_i^{e_i} \mid p_1^{f_1} \cdots p_k^{f_k}$ gdzie $\{p_i\}$ są różnymi liczbami pierwszymi i i 1. Wtedy $\varphi(N) = \prod_{i=1}^k (p_i - 1)$.

Przykład 8.20

Weźmy $N = 15 = 5 \cdot 3$. Następnie $\mathbb{Z} = \{1, 2, 4, 7, 8, 11, 13, 14\}$ i $|\mathbb{Z}| = 15 = 8 = 15 \cdot 4 \cdot 2 = \varphi(15)$. Odwrotnością 8 w \mathbb{Z} jest 2, ponieważ $8 \cdot 2 \equiv 1 \pmod{15}$.

Pokazaliśmy, że Z_N jest grupą rzędu $\varphi(N)$. Poniżej przedstawiono łatwe wnioski z Twierdzenia 8.14 i Wniosku 8.17:

DODATEK 8.21 Weź dowolną liczbę całkowitą $N > 1$ i a $\in Z_N$. Następnie

$$\varphi(N) = 1 \pmod{N}.$$

Dla konkretnego przypadku, gdy $N = p$ jest liczbą pierwszą i a $\in \{1, \dots, p-1\}$, mamy

$$a^{p-1} = 1 \pmod{\text{str.}}$$

DODATEK 8.22 Ustal $N > 1$. Dla liczby całkowitej $e > 0$ zdefiniuj $f_e : Z_N \rightarrow Z_N$ $[x \mapsto x^e]$. Jeśli e jest względnie pierwsze względem $\varphi(N)$, to f_e jest permutacją. Co więcej, jeśli $d = e^{-1} \pmod{\varphi(N)}$, to f_d jest odwrotnością f_e .

8.1.5 *Izomorfizmy i chińskie twierdzenie o resztach

Dwie grupy są izomorficzne, jeśli mają tę samą strukturę podstawową. Z matematycznego punktu widzenia izomorfizm grupy G zapewnia alternatywny, ale równoważny sposób myślenia o G. Z perspektywy obliczeniowej izomorfizm zapewnia inny sposób reprezentowania elementów w G, co często może mieć znaczący wpływ na efektywność algorytmiczną.

DEFINICJA 8.23 Niech G, H będą grupami odpowiednio ze względu na działania \cdot_G, \cdot_H . Funkcja $f : G \rightarrow H$ jest izomorfizmem od G do H jeśli:

1. f jest bijekcją i
2. Dla wszystkich $g_1, g_2 \in G$ mamy $f(g_1 \cdot_G g_2) = f(g_1) \cdot_H f(g_2)$.

Jeśli istnieje izomorfizm od G do H, to mówimy, że te grupy są izomorficzne i piszemy $G \cong H$.

W istocie izomorfizm z G do H jest po prostu zmianą nazwy elementów G na elementy H. Należy zauważyć, że jeśli G jest skończone i GH, to H musi być skończone i tej samej wielkości co G. Ponadto, jeśli istnieje izomorfizm f od G do H, wówczas f jest izomorfizmem od H do G. Jest jednak możliwe, że f można efektywnie obliczyć, podczas gdy f nie jest (lub odwrotnie).

Celem tej sekcji jest wykorzystanie języka izomorfizmów do lepszego zrozumienia struktury grupowej Z_N i Z , gdy $N = pq$ jest iloczynem dwóch różnych liczb pierwszych. Najpierw musimy wprowadzić pojęcie iloczynu bezpośredniego grup. Biorąc pod uwagę grupy G, H z operacjami grupowymi odpowiednio \cdot_G, \cdot_H , definiujemy nową grupę $G \times H$ (bezpośredni iloczyn $G \times H$) w następujący sposób. Elementy $G \times H$ są uporządkowanymi parami (g, h) z $g \in G$ i $h \in H$; zatem, jeśli G

ma n elementów, a H ma n elementów, $G \times H$ ma $n \cdot n$ elementów. Operacja grupowa na $G \times H$ jest stosowana składowo; to jest:

$$(g, h) \quad (g, h) \quad \stackrel{\text{def}}{=} (g \in G \text{ sol , godz } H \text{ godz}).$$

Pozostawiamy ćwiczeniu 8.8 sprawdzenie, czy $G \times H$ rzeczywiście jest grupą. Powyższą notację można w naturalny sposób rozszerzyć na bezpośrednie produkty z więcej niż dwóch grup, chociaż nie będzie nam to potrzebne w dalszej części.

Możemy teraz sformułować i udowodnić chińskie twierdzenie o resztach.

TWIERDZENIE 8.24 (Chiński twierdzenie o resztach) Niech $N = pq$ gdzie $p, q > 1$ są względnie pierwsze. Następnie

$$ZN \cong Zp \times Zq \text{ i } Z \cong Z_p \times Z_q.$$

Ponadto, niech f będzie funkcją odwzorowującą elementy $x \in \{0, \dots, N-1\}$ do par $(xp, xq) \in Zp \times Zq$ zdefiniowane przez

$$f(x) \stackrel{\text{def}}{=} ([x \bmod p], [x \bmod q]).$$

Następnie f jest izomorfizmem od ZN do $Zp \times Zq$ i ograniczeniem f do $Z \cong Z$ jest izomorfizmem z Z do $Z_p \times Z_q$.

DOWÓD Dla dowolnego $x \in ZN$ wyjściem $f(x)$ jest para elementów $(xp, xq) \in Zp \times Zq$. Twierdzimy, że jeśli $x \in Z$ to $(xp, xq) \in Z$ Rzeczywiście, jeśli $x \in Z$ $\Rightarrow p \nmid xp$ i $q \nmid xq$. To oznacza to, że $\gcd([x \bmod p], p) = 1$. Ale wtedy $\gcd(x, p) = 1$. To implikuje $\gcd(x, N) = 1$, co jest sprzeczne z założeniem, że $x \in Z$. Pokażemy teraz, że f jest izomorfizmem od ZN do $Zp \times Zq$. (Dowód podobny.) Zaczniemy od udowodnienia, że jest to izomorfizm $f(x) = (xp, xq) = f(x)$.
 \Rightarrow do $Z \cong Z$ izomorfizm z Z , że f jest jeden do jednego. Wtedy $x = xp = x \bmod p \Rightarrow x = xq = x \bmod q$. To z kolei oznacza, że $(x - x)$ jest podzielne zarówno przez p , jak i q . Ponieważ $\gcd(p, q) = 1$, Twierdzenie 8.4 mówi, że $pq = N$ dzieli $(x - x)$. Ale wtedy $x = x \bmod N$. Dla $x, x \in ZN$ oznacza to, że $x = x$, a więc f rzeczywiście jest jeden do jednego. Ponieważ $|ZN| = N = p \cdot q = |Zp| \cdot |Zq|$, rozmiary ZN i $Zp \times Zq$ są takie same. To w połączeniu z faktem, że f jest jeden do jednego, oznacza, że f jest bijektywne.

W kolejnym akapicie niech $+N$ oznacza dodawanie modulo N , a niech oznacza operację grupową w $Zp \times Zq$ (tj. dodawanie modulo p w pierwszym składniku i dodawanie modulo q w drugim składniku). Aby zakończyć dowód, że f jest izomorfizmem od ZN do $Zp \times Zq$, musimy pokazać, że dla wszystkich $a, b \in ZN$ zachodzi zasada, że $f(a +N b) = f(a) f(b)$.

Aby przekonać się, że to prawda, zauważ to

$$\begin{aligned} f(a + N b) &= [(a + N b) \bmod p], [(a + N b) \bmod q] \\ &= [(a + b) \bmod p], [(a + b) \bmod q] \\ &= [a \bmod p], [a \bmod q] & [b \bmod p], [b \bmod q] = f(a) f(b). \end{aligned}$$

(W przypadku drugiej równości powyżej używamy faktu, że $[(X \bmod N) \bmod p] = [[X \bmod p] \bmod p]$ gdy $p \mid N$; patrz ćwiczenie 8.9.) ■

Rozszerzenie chińskiego twierdzenia o resztach mówi, że jeśli p_1, p_2, \dots, p są parami względnie pierwsze (tzn. $\gcd(p_i, p_j) = 1$ dla wszystkich $i = j$) i $N \stackrel{\text{def}}{=} p_1 p_2 \cdots p_n$,

Następnie

$$ZN Z_{p_1} \times \cdots \times Z_{p_n} \quad N \quad Z_{p_1} \times \cdots \times Z \text{ str.}$$

Izomorfizm w każdym przypadku uzyskuje się poprzez naturalne rozszerzenie izomorfizmu użytego w powyższym twierdzeniu.

W notacji, przy zrozumieniu $N \in \{0, 1, \dots, N-1\}$ piszemy $x \equiv (xp, xq)$ dla $xp \equiv [x \bmod p]$ i $xq \equiv [x \bmod q]$. Oznacza to, że $x \equiv (xp, xq)$ wtedy i tylko wtedy, gdy $f(x) = (xp, xq)$, gdzie f jest takie jak w powyższym twierdzeniu. Można myśleć o tym zapisie w następujący sposób: oznacza on, że „ x (w Z_N) odpowiada (xp, xq) (w $Z_p \times Z_q$)”. Tę samą notację stosuje się w przypadku $x \in Z_N$.

Przykład 8.25

Weźmy $15 = 5 \cdot 3$ i rozważmy twierdzenie $15 = \{1, 2, 4, 7, 8, 11, 13, 14\}$. Chińczycy $\times Z_5$ o reszcie Z_3 , które mówi, że ta grupa jest izomorficzna z Z_3 . Możemy obliczyć

$$\begin{array}{ccccccccc} 1 & (1, 1) & 2 & (2, 2) & 4 & (4, 1) & 7 & (2, 1) & 8 & (3, 2) & 11 & (1, 2) \\ 13 & (3, 1) & 14 & (4, 2), \end{array}$$

gdzie każda para (a, b) z Z_5 oraz $b \in Z_3$ pojawia się dokładnie raz.

Korzystanie z chińskiego twierdzenia o resztach

Jeśli dwie grupy są izomorficzne, wówczas obie służą jako reprezentacje tej samej podstawowej „struktury algebraicznej”. Niemniej jednak wybór reprezentacji, którą należy zastosować, może mieć wpływ na wydajność obliczeniową operacji grupowych.

Omawiamy to abstrakcyjnie, a następnie w konkretnym kontekście Z_N i Z_3 .

Niech G, H będą grupami z operacjami odpowiednio \cdot_G , \cdot_H i powiedzmy, że f jest izomorfizmem od G do H , gdzie zarówno f , jak i f^{-1} można efektywnie obliczyć.

Następnie dla $g_1, g_2 \in G$ możemy obliczyć $g = g_1 \cdot_G g_2$ na dwa sposoby: albo bezpośrednio obliczając operację grupową w G , albo wykonując następujące kroki:

1. Oblicz $h_1 = f(g_1)$ i $h_2 = f(g_2)$;

2. Oblicz $h = h_1 \cdot h_2$ korzystając z operacji grupowej w H ;

3. Oblicz $g = f^{-1}(H)$.

Powyższe rozciąga się w naturalny sposób, gdy chcemy obliczyć wielokrotne operacje grupowe w G (np. obliczyć $g \cdot x$ dla pewnej liczby całkowitej x). To, która metoda jest lepsza, zależy od względnej wydajności obliczania działania grupowego w każdej grupie, a także wydajności obliczania f^{-1} .

Przejdzmy teraz do konkretnego przypadku obliczeń modulo N , gdy $N = pq$ jest iloczynem różnych liczb pierwszych. Chińskie twierdzenie o resztach pokazuje, że dodawanie, mnożenie lub potęgowanie (które jest po prostu powtarzanym mnożeniem) modulo N można „przekształcić” w analogiczne operacje modulo p i q . Opierając się na przykładzie 8.25, pokazujemy kilka prostych przykładów z $N = 15$.

Przykład 8.26

Załóżmy, że chcemy obliczyć iloczyn $14 \cdot 13$ modulo 15 (tj. w \mathbb{Z}_{15}). Przykład 8.25 daje $14 \equiv (4, 2)$ i $13 \equiv (3, 1)$ w $\mathbb{Z}_5 \times \mathbb{Z}_3$, mamy

$$(4, 2) \cdot (3, 1) = ([4 \cdot 3 \bmod 5], [2 \cdot 1 \bmod 3]) = (2, 2).$$

Uwaga $(2, 2) \equiv 2$, co jest poprawną odpowiedzią, ponieważ $14 \cdot 13 = 2 \bmod 15$.

Przykład 8.27

Załóżmy, że chcemy obliczyć $1153 \bmod 15$. Przykład 8.25 daje $11 \equiv (1, 2)$. Zauważ, że $2 \equiv 1 \bmod 3$ i tak

$$(1, 2)53 = ([153 \bmod 5], [(-1)53 \bmod 3]) = (1, [-1 \bmod 3]) = (1, 2).$$

Zatem $1153 \bmod 15 = 11$.

Przykład 8.28

Załóżmy, że chcemy obliczyć $[29100 \bmod 35]$. Najpierw obliczamy korelację $29 \equiv ([29 \bmod 5], [29 \bmod 7]) = ((1 \bmod 5), 1)$. Korzystając z chińskiego twierdzenia o resztach, mamy

$((1 \bmod 5), 1)100 = (((-1)100 \bmod 5), [1100 \bmod 7]) = (1, 1)$ i natychmiast widać, że $(1, 1) \equiv 1$. Mamy wynioskować, że $[29100 \bmod 35] = 1$.

Przykład 8.29

Załóżmy, że chcemy obliczyć $[1825 \bmod 35]$. Mamy $18 \equiv (3, 4)$ i tak $1825 \bmod$

$$35 \equiv (3, 4)25 = ([325 \bmod 5], [425 \bmod 7]).$$

Ponieważ \mathbb{Z}_5 jest grupą rzędu 4, możemy „przepracować modulo 4 w wykładniku” (por. Wniosek 8.15) i zobacz to

$$3^{25} = 3[25 \bmod 4] = 31 = 3 \bmod 5.$$

Podobnie,

$$4^{25} = 4[25 \bmod 6] = 41 = 4 \bmod 7.$$

Zatem $([325 \bmod 5], [425 \bmod 7]) = (3, 4) \quad 18$ i tak $[1825 \bmod 35] = 18$.

Jedną rzeczą, o której jeszcze nie rozmawialiśmy, jest to, jak dokonać konwersji pomiędzy reprezentacją elementu modulo N a jego reprezentacją modulo p i q. Konwersję można przeprowadzić wydajnie, pod warunkiem, że znana jest faktoryzacja N. Zakładając, że p i q są znane, łatwo jest odwzorować element x modulo N na jego odpowiednią reprezentację modulo p i q: element x odpowiada $([x \bmod p], [x \bmod q])$, a obydwie redukcje modułowe można przeprowadzić skutecznie (por. Załącznik B.2).

Dla drugiego kierunku korzystamy z następującej obserwacji: element z reprezentacją (xp, xq) można zapisać jako

$$(xp, xq) = xp \cdot (1, 0) + xq \cdot (0, 1).$$

Jeśli więc uda nam się znaleźć elementy $1p, 1q \in \{0, \dots, N-1\}$ tak, że $1p \equiv (1, 0)$ i $1q \equiv (0, 1)$, to (odwołując się do chińskiego twierdzenia o resztach) wiemy, że

$$(xp, xq) \equiv [(xp \cdot 1p + xq \cdot 1q) \bmod N].$$

Ponieważ p, q są różnymi liczbami pierwszymi, $\gcd(p, q) = 1$. Możemy użyć rozszerzonego algorytmu Eu-klidesa (por. Dodatek B.1.2), aby znaleźć liczby całkowite X, Y takie, że

$$Xp + Yq = 1.$$

Zauważ, że $Yq = 0 \bmod q$ i $Yq = 1 - Xp \equiv 1 \bmod p$. Oznacza to, że $[Yq \bmod N] \equiv (1, 0)$; tj. $[Yq \bmod N] = 1p$. Podobnie $[Xp \bmod N] = 1q$.

Podsumowując, możemy przekształcić element reprezentowany jako (xp, xq) na jego reprezentację modulo N w następujący sposób (zakładając, że znane są p i q):

1. Oblicz X, Y tak, że $Xp + Yq = 1$.
2. Ustaw $1p := [Yq \bmod N]$ i $1q := [Xp \bmod N]$.
3. Oblicz $x := [(xp \cdot 1p + xq \cdot 1q) \bmod N]$.

Jeśli zostanie wykonanych wiele takich konwersji, wówczas $1p, 1q$ można obliczyć raz na zawsze w fazie wstępnego przetwarzania.

Przykład 8.30

Weźmy $p = 5$, $q = 7$ i $N = 5 \cdot 7 = 35$. Założmy, że mamy reprezentację $(4, 3)$ i chcemy ją przekonwertować na odpowiedni element Z35. Korzystając z rozszerzonego algorytmu Euklidesa, obliczamy

$$3 \cdot 5 - 2 \cdot 7 = 1.$$

Zatem $1p = [2 \cdot 7 \bmod 35] = 21$ i $1q = [3 \cdot 5 \bmod 35] = 15$. (Możemy sprawdzić, czy są one poprawne: np. dla $1p = 21$ możemy sprawdzić, że $[21 \bmod 5] = 1$ i $[21 \bmod 7] = 0$.) Używając tych wartości, możemy następnie obliczyć

$$\begin{aligned}(4, 3) &= 4 \cdot (1, 0) + 3 \cdot (0, 1) \quad [4 \cdot \\ 1p + 3 \cdot 1q \bmod 35] &= [4 \cdot 21 \\ + 3 \cdot 15 \bmod 35] &= 24.\end{aligned}$$

Ponieważ $24 = 4 \bmod 5$ i $24 = 3 \bmod 7$, jest to rzeczywiście prawidłowy wynik.

8.2 Liczby pierwsze, faktoring i RSA

W tej sekcji pokazujemy pierwsze przykłady problemów z teorii liczb, które uważa się za „trudne”. Zaczynamy od omówienia jednego z najstarszych problemów: faktoryzacji liczb całkowitych lub po prostu faktoryzacji.

Mając złożoną liczbę całkowitą N , problem rozkładu na czynniki polega na znalezieniu takich liczb całkowitych $p, q > 1$, że $pq = N$. Rozkład na czynniki jest klasycznym przykładem trudnego problemu, zarówno dlatego, że jest łatwy do opisania, jak i dlatego, że został uznany za trudnym problemem obliczeniowym przez długi czas (nawet przed jego zastosowaniem w kryptografii). Problem można rozwiązać w czasie wykładniczym $O(N \cdot \text{polilog}(N))$ stosując dzielenie próbne: to znaczy wyczerpująco sprawdzając, czy p dzieli $N = 2, \dots, \text{polilog}(N)$. (Ta metoda wymaga N podziałów, przy czym każdy przyjmuje $p = N$ czas dla pewnej stałej c .) To zawsze się udaje, ponieważ chociaż największy czynnik pierwszy N może być tak duży jak $N/2$, najmniejszy czynnik pierwszy N może wynosić co najwyżej \sqrt{N} . Chociaż znane są algorytmy o lepszym czasie działania (patrz rozdział 9), pomimo wielu lat wysiłków nie zademonstrowano żadnego wielomianowego algorytmu rozkładu na czynniki.

Rozważmy następujący eksperyment dla danego algorytmu A i parametru n :

Eksperyment slabego faktoringu w-FactorA(n):

1. Wybierz dwie jednakowe n -bitowe liczby całkowite x_1, x_2 .
2. Oblicz $N := x_1 \cdot x_2$.
3. A ma dane N i wyniki x_1, x_2 . Wynik eksperymentu definiuje się jako 1, jeśli $x_1 \cdot x_2 = N$, i 0 w przeciwnym razie.

Powiedzieliśmy już, że problem faktoringu jest uważany za trudny. Robi to oznacza tamto

$$\Pr[w\text{-FactorA}(n) = 1] \text{ negl}(n) \text{ jest}$$

pomijalny dla każdego algorytmu ppt A? Zupełnie nie. Na początek liczba N w powyższym eksperymencie jest parzysta z prawdopodobieństwem $3/4$ (ma to miejsce, gdy

albo x_1 , albo x_2 jest parzyste); w tym przypadku A oczywiście łatwo jest rozłożyć N na czynniki. Chociaż możemy utrudnić zadanie A, wymagając od A wygenerowania liczb całkowitych o x_1, x_2 długości n, pozostaje faktem, że x_1 lub x_2 (a zatem N) mogą mieć małe czynniki pierwsze, które nadal można łatwo znaleźć. W przypadku zastosowań kryptograficznych będziemy musieli temu zapobiec.

Jak wskazuje ta dyskusja, „najtrudniej” rozłożyć liczby na czynniki, które mają tylko duże czynniki pierwsze. Sugeruje to przeddefiniowanie powyższego eksperymentu w taki sposób, aby x_1, x_2 były losowymi n-bitowymi liczbami pierwszymi, a nie losowymi n-bitowymi liczbami całkowitymi i faktycznie taki eksperiment zostanie zastosowany, gdy formalnie zdefiniujemy założenie dotyczące faktoringu w podrozdziale 8.2.3. Aby jednak ten eksperiment był użyteczny w zastosowaniach kryptograficznych, konieczna jest możliwość wydajnego generowania losowych n-bitowych liczb pierwszych. Jest to temat następnych dwóch rozdziałów.

8.2.1 Generowanie losowych liczb pierwszych

Naturalnym podejściem do generowania losowej n-bitowej liczby pierwszej jest wielokrotne wybieranie losowych n-bitowych liczb całkowitych, aż znajdziemy taką, która jest pierwsza; powtarzamy to co najwyżej t razy lub do momentu osiągnięcia sukcesu. Ogólny opis procesu można znaleźć w Algorytmie 8.31.

ALGORYTM 8.31

Generowanie losowej liczby pierwszej – zarys wysokiego poziomu

Wejście: Długość n; parametr t

Wynik: Jednolita n-bitowa liczba pierwsza

dla i = 1 do t: n - 1

p = {0, 1} p :=

1 p jeśli p

jest liczbą pierwszą, powrót

p nie powródł się

Należy zauważać, że algorytm wymusza, aby sygnał wyjściowy był liczbą całkowitą o długości dokładnie n (a nie o długości co najwyżej n), ustalając najbardziej znaczący bit p na „1”. Przyjęliśmy w tej książce konwencję, że „liczba całkowita o długości n” oznacza liczbę całkowitą, której reprezentacja binarna z najbardziej znaczącym bitem równym 1 ma długość dokładnie n bitów.

Mając możliwość ustalenia, czy dana liczba całkowita p jest liczbą pierwszą, powyższy algorytm generuje jednolitą n-bitową liczbę pierwszą uwarunkowaną przypadkiem, w którym nie zostanie wygenerowany błąd. Prawdopodobieństwo, że wyjścia algorytmu zawiodą, zależy od t i dla naszych celów będziemy chcieli ustawić t tak, aby uzyskać prawdopodobieństwo awarii, które jest zaniedbywalne w n. Aby pokazać, że algorytm 8.31 prowadzi do wydajnego (tj. wielomianowego czasu w n) algorytmu generowania liczb pierwszych, potrzebujemy lepszego zrozumienia dwóch kwestii: (1) prawdopodobieństwa, że jednolita n-bitowa liczba całkowita jest pierwsza oraz (2) jak skutecznie sprawdzić, czy dana liczba całkowita

p jest liczbą pierwszą. Omówimy teraz pokrótkie te kwestie, a bardziej szczegółowe zbadanie drugiego tematu odłożymy do następnej sekcji.

Rozkład liczb pierwszych. Twierdzenie o liczbach pierwszych, ważny wynik w matematyce, wyznacza dość dokładne granice ułamka liczb całkowitych o danej długości, które są liczbami pierwszymi. Dla naszych celów potrzebujemy jedynie słabej, jednostronnej wersji tego wyniku, którego tutaj nie udowadniamy:

TWIERDZENIE 8.32 (postulat Bertranda) Dla dowolnego $n > 1$ ułamek n -bitowych liczb całkowitych, które są pierwsze, wynosi co najmniej $1/3n$.

Wracając do opisanego powyżej podejścia do generowania liczb pierwszych, oznacza to, że jeśli ustalimy $t = 3n$, to prawdopodobieństwo, że liczba pierwsza nie zostanie wybrana we wszystkich t iteracjach algorytmu wynosi co najwyżej²

$$1 - \frac{1}{3n}^t = 1 - \frac{1}{3n}^{3n} \stackrel{n}{\text{mi}} 1 \text{ rz} = mi^n$$

(używając nierówności A.2), która jest zaniedbywalna w n . Zatem stosując iteracje $\text{poli}(n)$ otrzymujemy algorytm, dla którego prawdopodobieństwo niepowodzenia w wyniku jest znikome w n . (Znane są bardziej rygorystyczne wyniki niż w Twierdzeniu 8.32, dlatego w praktyce potrzebnych jest jeszcze mniej iteracji.)

Testowanie pierwszości. Problem skutecznego ustalenia, czy dana liczba jest pierwsza, ma długą historię. W latach 70. XX wieku opracowano pierwsze wydajne algorytmy testowania pierwszości. Algorytmy te były probabilistyczne i miały następującą gwarancję: jeśli na wejściu p byłoby liczbą pierwszą, algorytm zawsze wypisałby „pierwszą”. Z drugiej strony, jeśli p byłoby złożone, algorytm prawie zawsze wygenerowałby „złożony”, ale mógłby wypisać błędą odpowiedź („liczba pierwsza”) z prawdopodobieństwem znikomym na długości p . Inaczej mówiąc, jeśli algorytm daje wynik „złożony”, to p jest zdecydowanie złożone, ale jeśli wynikiem jest „pierwszy”, to jest bardzo prawdopodobne, że p jest liczbą pierwszą, ale jest też możliwe, że wystąpił błąd (a p jest w rzeczywistości złożone).

Podczas korzystania z tego rodzaju randomizowanego testu pierwszości w algorytmie 8.31 (algorytm generowania liczb pierwszych pokazany wcześniej), wynikiem algorytmu jest jednorodna liczba pierwsza o pożądanej długości, o ile algorytm nie zakończy się niepowodzeniem, a losowy test pierwszości nie zadziałał nie popełnić błędu podczas wykonywania algorytmu. Oznacza to, że wprowadzono dodatkowe źródło błędu (oprócz możliwości wprowadzenia błędu) i algorytm może teraz przez pomyłkę wypisać liczbę złożoną. Ponieważ możemy zapewnić, że stanie się to z znikomym prawdopodobieństwem, ta odległa możliwość nie ma praktycznego znaczenia i możemy ją bezpiecznie zignorować.

²Istnieją również probabilistyczne testy pierwszości, które działają w odwrotny sposób: zawsze poprawnie identyfikują liczby złożone, ale czasami popełniają błąd, gdy jako dane wejściowe podaje się liczbę pierwszą. Algorytmów tego typu nie będziemy rozważać.

Deterministyczny algorytm czasu wielomianowego do testowania pierwszości został zademonstrowany w przełomowym wyniku w 2002 roku. Algorytm ten, chociaż działa w czasie wielomianowym, jest wolniejszy niż wspomniane powyżej testy probabilistyczne. Z tego powodu probabilistyczne testy pierwszości są nadal stosowane wyłącznie w praktyce do generowania dużych liczb pierwszych.

W podrozdziale 8.2.2 opisujemy i analizujemy jeden z najczęściej stosowanych probabilistycznych testów pierwszości: algorytm Millera-Rabina. Algorytm ten pobiera dwa dane wejściowe: liczbę całkowitą p i parametr t (jednostkowy), który określa prawdopodobieństwo błędu. Algorytm Millera – Rabina działa w wielomianach czasu w p i t i spełnia:

TWIERDZENIE 8.33 Jeżeli p jest liczbą pierwszą, wówczas test Millera–Rabina zawsze daje w wyniku „liczbę pierwszą”. Jeśli p jest złożone, algorytm daje wynik „złożony”, chyba że prawdopodobieństwo wynosi co najwyżej 2^{-t} .

Kładąc wszystko razem. Biorąc pod uwagę poprzednią dyskusję, możemy teraz opisać algorytm generowania liczb pierwszych w czasie wielomianowym, który na wejściu n generuje n -bitową liczbę pierwszą, z wyjątkiem przypadku, gdy prawdopodobieństwo jest znikome w n ; ponadto, pod warunkiem, że wyjście p jest liczbą pierwszą, p jest n -bitową liczbą pierwszą o równomiernie rozłożonym rozkładzie. Pełną procedurę opisano w Algorytmie 8.34.

ALGORYTM 8.34 Generowanie

losowej liczby pierwszej

Wejście: Długość n

Wynik: Jednolita n -bitowa liczba pierwsza

2 dla $i = 1$ do $3n$:

$p \in \{0, 1\}^n$

1p

uruchom test Millera-Rabina na wejściu p i parametrze $1n$, jeśli wyjście jest „pierwsze”, powrót p powrót nie powiodł się

Generowanie liczb pierwszych określonej postaci. Czasami pożądane jest wygenerowanie losowej n -bitowej liczby pierwszej p określonej postaci, na przykład spełniającej $p = 3 \bmod 4$ lub takiej, że $p = 2q + 1$ gdzie q jest również liczbą pierwszą (o tego ostatniego typu nazywane są mocnymi liczbami pierwszymi). W takim przypadku można zastosować odpowiednie modyfikacje przedstawionego powyżej algorytmu generacji pierwotnej. (Na przykład, aby otrzymać liczbę pierwszą w postaci $p = 2q + 1$, zmodyfikuj algorytm tak, aby wygenerował losową liczbę pierwszą q , oblicz $p := 2q + 1$, a następnie wypisz p , jeśli również jest liczbą pierwszą.)

Chociaż te zmodyfikowane algorytmy sprawdzają się dobrze w praktyce, rygorystyczne dowody na to, że działają one w czasie wielomianowym i zawodzą z jedynie znikomym prawdopodobieństwem, są bardziej złożone (a w niektórych przypadkach opierają się na niepotwierdzonych domysłach teorii liczb

dotyczące gęstości liczb pierwszych określonej postaci). Szczegółowa eksploracja tych kwestii wykracza poza zakres tej książki i po prostu to założymy w razie potrzeby istnienie odpowiednich algorytmów pierwszej generacji.

8.2.2 *Testowanie pierwszości

Opiszymy teraz test pierwszości Millera-Rabina i udowodnimy Twierdzenie 8.33. (Opieramy się na materiale przedstawionym w rozdziale 8.1.5.) Ten materiał nie jest używany bezpośrednio w pozostałe części książki.

Kluczem do algorytmu Millera-Rabina jest znalezienie właściwości odróżniającej liczby pierwsze i złożone. Niech N oznacza numer wejścia, które ma zostać przetestowane.

Zaczynamy od następującej obserwacji: jeśli N jest liczbą pierwszą, to $|Z| = N - 1$, i tak dla dowolnego $a \in \{1, \dots, N-1\}$ mamy $a^{N-1} \equiv 1 \pmod{N}$ zgodnie z twierdzeniem 8.14.

Sugeruje to sprawdzenie, czy N jest liczbą pierwszą, wybierając jednolity element a

i sprawdzenie, czy $a^{N-1} \not\equiv 1 \pmod{N}$. Jeśli $a^{N-1} \equiv 1 \pmod{N}$, to N nie może być liczbą pierwszą. I odwrotnie, możemy mieć nadzieję, że jeśli N nie jest liczbą pierwszą, to tak jest rozsądna szansa, że wybierzemy z $a^{N-1} \not\equiv 1 \pmod{N}$ i tak powtarzając ten test wiele razy, możemy określić, czy N jest liczbą pierwszą, czy nie z dużą pewnością siebie. Powyższe podejście przedstawiono jako algorytm 8.35.

(Przypomnijmy sobie potęgowanie modulo N i obliczanie największej wspólnej dzielniki można przeprowadzić w czasie wielomianowym. Wybór jednolitego elementu z $\{1, \dots, N-1\}$ można również wykonać w czasie wielomianowym. Patrz dodatek B.2.)

ALGORYTM 8.35

Testowanie pierwszości – pierwsza próba

Wejście: liczba całkowita N i parametr $1t$

Wynik: decyzja, czy N jest liczbą pierwszą, czy złożoną

dla $i = 1$ do t :

za $\{1, \dots, N-1\}$

Jeśli $a^{N-1} \equiv 1 \pmod{N}$ zwraca „kompozyt”

zwrócić „pierwszy”

Jeśli N jest liczbą pierwszą, algorytm zawsze zwraca wartość „pierwsza”. Jeśli N jest złożone, to algorytm generuje wynik „złożony”, jeśli w dowolnej iteracji znajdzie $a \in \{1, \dots, N-1\}$ taki, że $a^{N-1} \not\equiv 1 \pmod{N}$. Zauważ, że jeśli $a \in \{1, \dots, N-1\}$ i $a^{N-1} \not\equiv 1 \pmod{N}$, to $\gcd(a, N) > 1$. (Jeśli $\gcd(a, N) = 1$, to $\gcd(a^{N-1}, N) = 1$ i tak $[a^{N-1}]^{\frac{1}{\gcd(a, N)}} \equiv 1 \pmod{N}$ nie może być równego 1.) Dlatego na razie ograniczamy naszą uwagę do Z_N . Do każdego takiego słowa odnosimy się za pomocą $a^{N-1} \equiv 1 \pmod{N}$ jako świadek, że N jest złożony, lub po prostu świadek. Możemy mieć nadzieję, że gdy N jest złożone, jest ich wiele świadków, dlatego algorytm znajduje takiego świadka z „wysokim” prawdopodobieństwem. Ta intuicja jest słuszna pod warunkiem, że znajdzie się chociaż jeden świadek. Przed udowodnieniem potrzebujemy dwóch lematów z teorii grup.

TWIERDZENIE 8.36 Niech G będzie grupą skońzoną, a $H \subseteq G$. Załóżmy, że H jest niepuste i dla wszystkich $a, b \in H$ mamy $ab \in H$. Wtedy H jest podgrupą G .

DOWÓD Musimy sprawdzić, czy H spełnia wszystkie warunki Definicji 8.9. Z założenia H jest domknięty w ramach operacji grupowej. Łączność w H jest automatycznie dziedziczona z G . Niech $m = |G|$ (w tym miejscu wykorzystujemy fakt, że G jest skończony) i rozważamy dowolny element $a \in H$. Zamknięcie $m^{-1}H$ oznacza, że H zawiera $a = 1$. Zatem H zawiera odwrotność każdego ze swoich elementów, jak i tożsamość, jak również m^{-1} . ■

LEMAT 8.37 Niech H będzie ścisłą podgrupą skończonej grupy G (tzn. $H = G$).

Następnie $|H| \leq |G|/2$.

DOWÓD Niech h^- będzie elementem G , którego nie ma w H ; ponieważ $H = G$, wiemy, że takie h^- istnieje. Rozważmy zbiór $H^- = \{hh^- \mid \text{godz } h \in H\}$. Pokazujemy, że (1) $|H^-| = |H|$, oraz (2) każdy element H^- leży poza H ; tj. przecięcie $H \cap H^-$ jest puste. Ponieważ zarówno H , jak i H^- są podzbiorami G , implikują one $|G| = |H| + |H^-| = 2|H|$, dowodząc lematu.

Dla dowolnego $h_1, h_2 \in H$, jeśli $hh_1^- = hh_2^-$ wtedy, mnożąc przez h^- z każdej strony, otrzymujemy $h_1 = h_2$. To pokazuje, że każdemu odrębnemu elementowi $h \in H$ odpowiada odrębnemu elementowi $hh^- \in H^-$, dowodząc (1).

Załóżmy w kierunku sprzecznosci, że $hh^- \in H$ dla pewnego h . Oznacza to $hh^- = h$ dla pewnego $h \in H$, a więc $h^- = hh \in H$, ponieważ $h \in H$ jest podgrupą $h^{-1}H$. Ale to oznacza, że $h = h^- \in H$, w przeciwnieństwie do h^- zostało wybrane. To dowodzi (2) i kończy dowód lematu. ■

Poniższe twierdzenie umożliwi nam analizę podanego wcześniej algorytmu.

TWIERDZENIE 8.38 Napraw N. Załóżmy, że istnieje świadek, że N jest złożone.

Wtedy co najmniej połowa elementów Z jest świadkami tego, że N jest złożony.

DOWÓD Niech Bad będzie zbiorem elementów w $Z \setminus N$, czyli $a \in Bad$ którzy nie są świadkami; $= 1$. Bad oznacza $a \in (ab)^{N-1} \mod N$. Oczywiście, $1 \in Z \setminus N$. Jeśli $a, b \in Z \setminus N$, $= 1 \cdot 1 = 1 \in N$. Ponieważ (z założenia)

Lemat 8.36 stwierdzamy, że Bad jest podgrupą Z

istnieje co najmniej jeden świadek, $Z \setminus N$ jest ścisłą podgrupą Z , a następnie $|Bad| \geq |Z \setminus N|/2$. Lemat 8.37 $|Bad| \geq |Z \setminus N|/2$ pokazuje, że $|Z \setminus N|/2$ nie są w N pokazujący, że co najmniej połowa elementów $Z \setminus N$ znajdują się w N (i dlatego są świadkami). ■

Niech N będzie złożone. Jeśli istnieje świadek, że N jest złożone, to $|Bad| \geq |Z \setminus N|/2$ świadków. Istnieje co najmniej $|Z \setminus N|/2$ prawdopodobieństwo, że w dowolnej iteracji algorytmu znajdziemy świadek lub element, którego nie ma w $Z \setminus N$ albo a , wynosi

zatem co najmniej $1/2$, a zatem prawdopodobieństwo, że algorytm nie znajdzie świadka w żadnej z iteracji (a zatem prawdopodobieństwo, że algorytm błędnie wypisze „pierwszą”) wynosi co najwyżej 2^{-t} .

Powyższe niestety nie daje pełnego rozwiązania, ponieważ istnieje nieskończenie wiele liczb złożonych N , które nie mają żadnych świadków, że są złożone! Takie wartości N są znane jako liczby Carmichaela; szczegółowe omówienie wykracza poza zakres tej książki.

Na szczęście można wykazać, że udoskonalenie powyższego testu działa dla wszystkich N . Niech $N \equiv 1 \pmod{2r}$, gdzie r jest nieparzyste, a $r \geq 1$. (Łatwo jest obliczyć r i u , mając N). Ponadto ograniczenie do $r \geq 1$ oznacza, że N jest nieparzyste, ale sprawdzenie pierwszości jest łatwe, gdy N jest parzyste!) Algorytm pokazany wcześniej sprawdza tylko, czy $1 \pmod{N}$ jest równa $a^{N-1} \equiv a^2 \pmod{ru}$. Bardziej wyrafinowany algorytm sprawdza sekwencję $a^2 \pmod{ru}, (a^2)^2 \pmod{ru}, \dots, (a^2)^{r-1} \pmod{ru}$. Każdy wyraz w tej sekwencji jest kwadratem poprzedniego wyrazu; zatem jeśli jakaś wartość jest równa ± 1 , to wszystkie kolejne wartości będą równe 1.

Powiedzmy, że $a \in Z$ jest mocnym dowodem na to, że N jest złożony (lub po prostu silnym świadkiem 2), jeśli (1) $a \equiv \pm 1 \pmod{N}$ i (2) $a \equiv -1 \pmod{N}$ dla wszystkich $i \in \{1, \dots, r-1\}$. Zauważ, że gdy element a nie jest mocnym świadkiem $a^2 \pmod{ru}$, wówczas ciąg $(a^i \pmod{ru})$ (wszystkie wzięte modulo N) przyjmuje jedną z następujących postaci:

$$(\pm 1, 1, \dots, 1) \text{ lub } (-1, 1, \dots, 1),$$

gdzie jest dowolnym terminem. Jeśli a nie jest mocnym świadkiem, to mamy $a^2 \pmod{ru} \equiv \pm 1 \pmod{N}$

$$N-1 \equiv a^2 \pmod{ru} \quad 2r-1 \equiv a^2 \pmod{u} = 1 \pmod{N},$$

a więc a nie jest także świadkiem tego, że N jest złożone. Inaczej mówiąc, jeśli świadek jest świadkiem, to jest także mocnym świadkiem, zatem mocnych świadków może być tylko więcej niż świadków.

Najpierw pokażemy, że jeśli N jest liczbą pierwszą, to nie istnieje mocny dowód na to, że N jest złożone. Czytając to, opieramy się na następującym łatwym lematce (który jest szczególnym przypadkiem Twierdzenia 13.16 udowodnionego później w Rozdziale 13):

LEMMA 8.39 Powiedzmy, że $x \in Z$ jest pierwiastkiem kwadratowym z 1 modulo N jeśli $x^2 \equiv 1 \pmod{N}$. Jeśli N jest liczbą pierwszą nieparzystą, to jedynymi pierwiastkami kwadratowymi z 1 modulo N są $\{\pm 1 \pmod{N}\}$.

DOWÓD Powiedzmy $x^2 \equiv 1 \pmod{N}$ z $x \in \{1, \dots, N-1\}$. Wtedy $0 \equiv x(x+1)(x-1) \pmod{N}$, co oznacza, że $N \mid (x+1)$ lub $N \mid (x-1)$ zgodnie z Twierdzeniem 8.3. Może to nastąpić tylko wtedy, gdy $x = \{\pm 1 \pmod{N}\}$. ■

Niech N będzie nieparzystą liczbą pierwszą i ustali dowolne $a \in Z \setminus N$. Niech $i = 0$ będzie minimum = mod N ; ponieważ $a^2 \pmod{ru}$ dla której $a = 1 \pmod{N-1} = a^2 \pmod{N}$. Wiemy o tym

istnieje takie $i \in \mathbb{Z}$. Jeśli $i = 0$, to świadek. Wtedy $a^{ty} = 1 \pmod{N}$ i a nie jest mocny przeciwnym razie,

$$a^{2(i-1)u} = a^{2iu} = 1 \pmod{N}$$

i a^{2iu} jest pierwiastkiem kwadratowym z 1. Jeśli N jest nieparzystą liczbą pierwszą, jedyne pierwiastki kwadratowe z $2^i u^2$ to ± 1 wynoszą ± 1 ; jednakże przez wybór $i \in \mathbb{Z}$ mamy $a = -1 \pmod{N}$. Zatem $a = -1 \pmod{N}$ i a nie jest mocnym świadkiem. Dochodzimy do wniosku, że gdy N jest nieparzystą liczbą pierwszą, nie ma mocnego dowodu na to, że N jest złożone.

Złożona liczba całkowita N jest potęgą pierwszą, jeśli $N = p^r$ dla pewnej liczby pierwszej p i liczby całkowitej $r \geq 1$. Pokażemy teraz, że każde nieparzyste, złożone N , które nie jest potęgą pierwszą, ma wielu mocnych świadków.

TWIERDZENIE 8.40 Niech N będzie liczbą nieparzystą, która nie jest potągą pierwszą. Zatem co najmniej połowa elementów Z_N jest mocnym dowodem na to, że N jest złożone.

DOWÓD Niech $\text{Bad} = \{a \in \mathbb{Z}_N \mid a \text{ nie jest mocnym świadkiem}\}$. Definiujemy zbiór Bad i pokazujemy, że: (1) Bad jest podzbiorem \mathbb{Z}_N i (2) Bad jest ścisłą podgrupą \mathbb{Z}_N . To wystarczy, ponieważ łącząc $|/2|$. Ponadto przez (1) to $|/2|$ jak w (2) i Lemat 8.37 mamy to $|\text{Zły}| \leq |\mathbb{Z}_N|$ utrzymuje, że $|\text{Zły}| \leq |\mathbb{Z}_N|$ Twierdzeniu 8.38. są mocnymi świadkami. (Podkreślamy to, i tak $|\text{Zły}| \leq |\mathbb{Z}_N|$ świadkami.) Zatem co najmniej połowa elementów Z_N nie są mocnymi świadkami. (Podkreślamy to, i tak $|\text{Zły}| \leq |\mathbb{Z}_N|$ świadkami.) Twierdzimy, że Bad jest podgrupą \mathbb{Z}_N . Najpierw zauważmy, że Bad jest podzbiorem \mathbb{Z}_N .

zauważmy, że $1 \in \text{Bad}$, ponieważ $(-1)^u = -1 \pmod{N}$ (przypomnijmy, że u jest nieparzyste). Niech $i \in \{0, \dots, r-1\}$ będzie największą liczbą całkowitą, dla której istnieje $a \in \text{Bad}$ with $a^{2iu} = 1 \pmod{N}$; alternatywnie i jest największą liczbą całkowitą, dla której istnieje $a \in \text{Bad}$ with $a^{2iu} = 1 \pmod{N}$.

$$(a^{ty}, a^{2u}, \dots, a^{2ru}) = (\underbrace{1, \dots, 1}_{i+1 \text{ warunki}}, \dots, 1)$$

Ponieważ $-1 \in \text{Zły}$ i $(-1)^0 = 1 \pmod{N}$, istnieje takie i .

Napraw i jak powyżej i zdefiniuj

$$\text{Zły} \stackrel{\text{def}}{=} \{a \mid a^2 \equiv 1 \pmod{N}\}.$$

Udowodnimy teraz to, co twierdziliśmy powyżej.

ZASTRZEŻENIE 8.41 $\text{Zły} \subseteq \mathbb{Z}_N$.

czerwca Niech $\text{Bad} = \{a \mid a^2 \equiv 1 \pmod{N} \text{ albo } a = 1 \pmod{N} \text{ dla } \text{około } 2 \text{ i } a \not\equiv 1 \pmod{N}\}$. W pierwszym przypadku $a = (a \cdot u) = 1 \pmod{N}$ i tak $a \in \text{Bad}$. W drugim przypadku mamy $j \in \mathbb{Z}$ i poprzez wybór i . Jeżeli $j = i$ to oczywiście $a \in \text{Bad}$.

< i to $a = (\text{dowolne}, \text{to pokazuje})$ jeśli $j^2 \equiv 1 \pmod{N}$ i $a \in \text{Bad}$. Ponieważ był
 $\text{Bad} \subset \text{Bad}$.

Twierdzenie 8.42 Bad jest podgrupą Z_N .

Oczywiście 1. Zły. Ponadto, jeśli $a, b \in \text{Zły}$, to

$$jm(ab)^2 \equiv za^2 \equiv b^2 \equiv (\pm 1)(\pm 1) \equiv \pm 1 \pmod{N}$$

i tak $ab \in \text{Zły}$. Według Lematu 8.36 Bad jest podgrupą.

Twierdzenie 8.43 Bad jest ścisłą podgrupą Z_N .

Jeśli N jest nieparzystą złożoną liczbą całkowitą, która nie jest potęgą pierwszą, wówczas N można zapisać jako $N = N_1 N_2$ z $N_1, N_2 > 1$ nieparzystym i $\gcd(N_1, N_2) = 1$. Odwołując się do chińskiego twierdzenia o resztach, niech $a = (a_1, a_2)$ oznaczają reprezentację N_2 ; to znaczy $a_1 = 1$ takie, że $a = (a_1, a_2)$ jako element $Z_{N_1} \times Z_{N_2}$ [a mod N_1] i $a_2 = a \pmod{N_2}$. Weź $b \in \text{Bad}$
 $1 \pmod{N}$ (takie a musi istnieć, jak zdefiniowaliśmy) i powiedz $a = (a_1, a_2)$. Ponieważ $-1 \pmod{-1}$, mamy

$$(a_1, a_2)^{2j.m} = (a_1^{2j.m}, a_2^{2j.m}) = (-1, -1),$$

a więc

$$2 \pmod{N} \equiv 1 \pmod{N_1} \text{ i } a = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

Rozważmy element $b \in Z_N$ z $b \equiv (a_1, 1)$. Następnie

$$b^{2j.m} = (a_1, 1)^{2j.m} \equiv (a_1 \pmod{N_1}, 1) = (-1, 1) \equiv \pm 1.$$

Oznacza to, $\pm 1 \pmod{N}$ i tak znaleźliśmy element $b \in \text{Bad}$. To i tak, zgodnie z Lematem 8.37, że b dowodzi, że Bad jest ścisłą podgrupą Z_N (a Z_N rozmiar zatem rozmiar Bad) jest co najwyżej połowę mniejszy od Z_N . ■

Liczba całkowita N jest potęgą doskonałą, jeśli $N = N^\wedge$ dla liczb całkowitych N^\wedge i $e \geq 2$ (tutaj nie jest wymagane, aby N^\wedge było liczbą pierwszą, chociaż oczywiście każda potęga pierwsza jest również potęgą doskonałą). Algorytm 8.44 daje test pierwszości Millera-Rabina. W Ćwiczeniach 8.12 i 8.13 masz pokazać, że sprawdzanie, czy N jest potęgą doskonałą i sprawdzanie, czy konkretne a jest mocnym świadkiem, można przeprowadzić w czasie wielomianowym. Biorąc pod uwagę te wyniki, algorytm wyraźnie działa w wielomianu czasu w N i t . Możemy teraz zakończyć dowód Twierdzenia 8.33:

DOWÓD Jeśli N jest nieparzystą liczbą pierwszą, nie ma mocnych świadków, dlatego algorytm Millera-Rabina zawsze zwraca „liczbę pierwszą”. Jeśli N jest parzyste lub jest potęgą pierwszą, algorytm zawsze daje wynik „złożony”. Ciekawy przypadek ma miejsce, gdy N jest nieparzystą złożoną liczbą całkowitą, która nie jest potęgą pierwszą. Rozważ dowolny

ALGORYTM 8.44 Test

pierwszości Millera–Rabina

Wejście: liczba całkowita $N > 2$ i parametr $1t$

Wynik: decyzyja, czy N jest liczbą pierwszą, czy złożoną

jeśli N jest parzyste, zwróć „złożone”,

jeśli N jest potęgą doskonałą, zwróć „złożone”

obliczenie $r = 1$ i u nieparzyste tak, że $N - 1 = 2ru$ dla j

$= 1$ do t : a

$\{1, \dots, N - 1\}$ jeśli a

$$y^{2^iu} = \pm 1 \pmod{N} \text{ i } a^{2^iu} = 1 \pmod{N} \text{ dla } i \in \{1, \dots, r - 1\}$$

zwróć „złożony” zwróć

„pierwszy”

iteracja pętli wewnętrznej. Zauważ najpierw, że jeśli $a^{2^iu} = 1 \pmod{N}$ i $a^{2^iu} \neq 1 \pmod{N}$ dla $i \in \{1, \dots, r - 1\}$. Prawdopodobieństwo znalezienia któregokolwiek a wynosi mocny świadek lub element spoza Z . Zatem N co najmniej $1/2$ (przywołując Twierdzenie 8.40). prawdopodobieństwo, że algorytm nigdy nie wygeneruje wartości „złożonej” w żadnej z iteracji, wynosi co najwyżej 2^{-t} .

8.2.3 Założenie faktoringowe

Niech GenModulus będzie algorytmem działającym w czasie wielomianowym, który na wejściu $1n$ daje wyniki (N, p, q) , gdzie $N = pq$, a p i q są n-bitowymi liczbami pierwszymi, z wyjątkiem prawdopodobieństwa zaniedbywalnego w n . (Naturalnym sposobem na osiągnięcie tego jest wygenerowanie dwóch jednakowych n-bitowych liczb pierwszych, jak omówiono wcześniej, a następnie pomnożenie ich w celu uzyskania N .) Następnie rozważ następujący eksperyment dla danego algorytmu A i parametru

Eksperyment faktoringowy FactorA,GenModulus(n):

1. Uruchom GenModulus($1n$), aby otrzymać (N, p, q) .
2. A ma dane N i wyprowadza $p, q > 1$.
3. Wynik eksperymentu definiuje się jako 1, jeśli $p \cdot q = N$, i 0 w przeciwnym razie.

Zauważ, że jeśli wynik eksperymentu wynosi 1, to $\{p, q\} = \{p, q\}$, chyba że p lub q są złożone (co zdarza się z znakomitym prawdopodobieństwem).

Teraz formalnie definiujemy założenie faktoringu:

DEFINICJA 8.45 Rozkładanie na czynniki jest trudne w stosunku do GenModulus, jeśli dla wszystkich probabilistycznych algorytmów A w czasie wielomianowym istnieje pomijalna funkcja negl taka, że

$$\Pr[WspółczynnikA, \text{GenModulus}(n) = 1] = \text{negl}(n).$$

Założeniem faktoringu jest założenie, że istnieje GenModulus w stosunku do czego faktoring jest trudny.

8.2.4 Założenie RSA

Problem faktoringu był badany przez setki lat i nie znaleziono skutecznego algorytmu. Chociaż założenie faktoringu faktycznie daje funkcję jednokierunkową (patrz sekcja 8.4.1), niestety nie daje bezpośrednio praktycznych kryptosystemów. (Jednak w podrozdziale 13.5.2 pokazujemy, jak konstruować wydajne kryptosystemy w oparciu o problem, którego trudność jest równoważna trudnością faktoringu.) To motywowało do poszukiwania innych problemów, których trudność jest związana z trudnością faktoringu. Najbardziej znanym z nich jest problem wprowadzony w 1978 roku przez Rivesta, Shamira i Adlemana i obecnie nazwany na ich cześć problemem RSA.

Biorąc pod uwagę moduł N i liczbę całkowitą $e > 2$ względnie pierwszą względem $\varphi(N)$, wniosek z rachunku 8.22 pokazuje, że potęgowanie do modułu potęgi e jest permutacją. Możemy zatem zdefiniować $[y \cdot e \text{ mod } N]$ (dla dowolnego $y \in Z_{\varphi(N)}$) jako wyjątkowy element $Z_{\varphi(N)}$ które daje y po podniesieniu do potęgi e modulo N , czyli $= y \text{ mod } N$.
 $x = y \cdot e \text{ mod } N$ wtedy i tylko wtedy, gdy $x^{\varphi(N)} = y \text{ mod } N$. Problem RSA, nieformalnie, polega na obliczeniu $[y \cdot e \text{ mod } N]$ dla modułu N o nieznanej faktoryzacji.

Formalnie, niech GenRSA będzie probabilistycznym algorytmem wielomianowym w czasie, który na wejściu $1n$ wyrowadza moduł N będący iloczynem dwóch n -bitowych liczb pierwszych, a także liczb całkowitych $e, d > 0$ z $\gcd(e, \varphi(N)) = 1$ i $ed = 1 \text{ mod } \varphi(N)$. (Takie ogłoszenie istnieje, ponieważ e jest odwracalne modulo $\varphi(N)$. Cel d stanie się jasny później.) Algorytm może zawieźć z znikomym prawdopodobieństwem w n . Rozważmy następujący eksperyment dla danego algorytmu A i parametru n :

Eksperyment RSA RSA-invA,GenRSA(n):

1. Uruchom GenRSA($1n$), aby uzyskać (N, e, d) .
2. Wybierz dowolny $y \in Z_{\varphi(N)}$.
3. A ma dane N, e, y i wyniki $x \in Z_{\varphi(N)}$.
4. Wynik eksperymentu definiuje się jako 1, jeśli $x^{\varphi(N)} = y \text{ mod } N$, i 0 w przeciwnym razie.

DEFINICJA 8.46 Problem RSA jest trudny w porównaniu z GenRSA, jeśli dla wszystkich probabilistycznych algorytmów A w czasie wielomianowym istnieje pomijalna funkcja negl taka, że $\Pr[\text{RSA-invA,GenRSA}(n) = 1] = \text{negl}(n)$.

Założeniem RSA jest to, że istnieje algorytm GenRSA, w stosunku do którego problem RSA jest trudny. Odpowiedni algorytm GenRSA można skonstruować z dowolnego algorytmu GenModulus, który generuje moduł złożony wraz z jego faktoryzacją. Ogólny zarys przedstawiono w postaci algorytmu 8.47, w którym jedną rzeczą nieokreślona jest to, w jaki sposób dokładnie wybierane jest e . W rzeczywistości uważa się, że problem RSA jest trudny dla dowolnego e , które jest względnie pierwsze względem $\varphi(N)$.

Poniżej omówimy kilka typowych wyborów e .

ALGORYTM 8.47 GenRSA –
zarys wysokiego poziomu

Wejście: Parametr bezpieczeństwa $1n$
Wyjście: N, e, d zgodnie z opisem w tekście

$(N, p, q) \quad \text{GenModulus}(1n)$
 $\varphi(N) := (p - 1)(q - 1)$
wybierz $e > 1$ takie, że $\gcd(e, \varphi(N)) = 1$ oblicz $d := [e^{-1} \bmod \varphi(N)]$ powrót N, e, d

Przykład 8.48

Załóżmy, że wyjścia GenModulus $(N, p, q) = (143, 11, 13)$. Następnie $\varphi(N) = 120$. Następnie musimy wybrać e , które jest względnie pierwsze względem $\varphi(N)$; powiedzmy, że bierzemy $e = 7$. Następnym krokiem jest obliczenie d w taki sposób, że $d = [e^{-1} \bmod \varphi(N)]$. Można to zrobić w sposób pokazany w dodatku B.2.2, aby otrzymać $d = 103$. (Można sprawdzić, że $7 \cdot 103 = 721 = 1 \bmod 120$.) Nasz algorytm GenRSA daje w ten sposób wynik $(143, 7, 103)$.

Jako przykład problemu RSA w odniesieniu do tych parametrów, weź $y = 64$, a zatem problemem jest obliczenie 7. pierwiastka z 64 modulo 143 bez znajomości d lub rozkładu na czynniki N .

Obliczanie pierwiastków eth modulo N staje się łatwe, jeśli znane jest $d, \varphi(N)$ lub rozkład na czynniki N . (Jak pokażemy w następnej sekcji, każdą z nich można wykorzystać do wydajnego obliczenia pozostałych). Wynika to z Wniosku 8.22, który pokazuje, że $[y \bmod N]$ jest pierwiastkiem eth z y modulo N . Ta asymetria —mianowicie że problem RSA wydaje się trudny, gdy nie jest znane d lub rozkład na czynniki N , ale staje się łatwy, gdy znane jest d —służy jako podstawa zastosowań problemu RSA w kriptografii klucza publicznego.

Przykład 8.49

Kontynuując poprzedni przykład, możemy obliczyć pierwiastek 7. z 64 modulo 143, używając wartości $d = 103$; odpowiedź brzmi $25 = 64d = 64103 \bmod 143$. Możemy sprawdzić, czy jest to prawidłowe rozwiązanie, ponieważ $25^7 = 257 = 64 \bmod 143$.

O wyborze E . Nie wydaje się, aby istniała jakakolwiek różnica w twardości problemu RSA dla różnych wykładowników e , w związku z czym zaproponowano różne metody jego wyboru. Jednym z popularnych wyborów jest ustalenie $e = 3$, ponieważ wtedy obliczenie potęgi eth modulo N wymaga tylko dwóch mnożeń (patrz Dodatek B.2.3). Jeśli e ma być równe 3, wówczas p i q należy wybrać za pomocą $p, q = 1 \bmod 3$ tak, aby $\gcd(e, \varphi(N)) = 1$. Z podobnych powodów innym popularnym wyborem jest $e = 216 + 1 = 65537$, liczba pierwsza o niskiej wadze Hamminga (w dodatku B.2.3 wyjaśniamy, dlaczego takie wykładowniki są preferowane). W porównaniu do wyboru $e = 3$, powoduje to potęgowanie

nieco droższe, ale zmniejsza ograniczenia na p i q i pozwala uniknąć niektórych „ataków o niskim wykładniku” (opisanych na końcu sekcji 11.5.1), które mogą wynikać ze źle zaimplementowanych kryptosystemów opartych na problemie RSA.

Zauważ, że wybór d small (to znaczy zmiana GenRSA na małe d , a następnie obliczenie $e := [d \bmod \varphi(N)]$) jest złym pomysłem. Jeśli d leży w bardzo małym zakresie, można przeprowadzić wyszukiwanie d metodą brute-force (i, jak zauważono, gdy d jest znane, problem RSA można łatwo rozwiązać). Nawet jeśli d zostanie wybrane w taki sposób, że $d \equiv N^{1/4}$, a zatem ataki typu brute-force zostaną wykluczone, istnieją znane algorytmy, których można w tym przypadku użyć do odzyskania d z N i e .

8.2.5 *Powiązanie założeń RSA i faktoringu

Załóżmy, że GenRSA jest skonstruowany jak w algorytmie 8.47. Jeśli N można rozłożyć na czynniki, możemy obliczyć $\varphi(N)$ i użyć tego do obliczenia $d := [e \bmod \varphi(N)]$ dla dowolnego e (używając Algorytmu B.11). Zatem, aby problem RSA był trudny w porównaniu z GenRSA, problem faktoringu musi być trudny w porównaniu z GenModulus. Inaczej mówiąc, problem RSA nie może być trudniejszy niż faktoring; trudność faktoringu (w stosunku do GenModulus) może być jedynie potencjalnie słabszym założeniem niż trudność problemu RSA (w stosunku do GenRSA).

A co z drugim kierunkiem? To znaczy, czy trudność problemu RSA wynika z trudności faktoringu? To pozostaje kwestią otwartą. Jedyne, co możemy pokazać, to to, że obliczenie d z N i e jest tak samo trudne jak rozkład na czynniki.

TWIERDZENIE 8.50 Istnieje probabilistyczny algorytm wielomianowy w czasie, który, mając na wejściu złożoną liczbę całkowitą N i liczby całkowite e , d gdzie $ed = 1 \bmod \varphi(N)$, daje na wyjściu współczynnik N , z wyjątkiem prawdopodobieństwa zaniedbywalnego w N .

DOWÓD Twierdzenie obowiązuje dla dowolnego N , ale dla uproszczenia – a ponieważ jest to przypadek najbardziej odpowiedni dla kryptografii – skupiamy się tutaj na przypadku, gdy N jest iloczynem dwóch różnych (nieparzystych) liczb pierwszych. Opieramy się na Twierdzeniu 8.36 i Lemacie 8.37, a także na następujących faktach (które wynikają z bardziej ogólnych wyników udowodnionych w rozdziałach 13.4.2 i 13.5.2):

- Jeśli N jest iloczynem dwóch różnych, nieparzystych liczb pierwszych, to 1 ma dokładnie cztery pierwiastki kwadratowe modulo N . Dwa z nich to „trywialne” pierwiastki kwadratowe ± 1 , a dwa z nich to „nietrywialne” pierwiastki kwadratowe.
- Do (efektywnego) obliczenia współczynnika N można zastosować dowolny nietrywialny pierwiastek kwadratowy z 1. Dzieje się tak dzięki temu, że $y^2 \equiv 1 \bmod N$ implikuje

$$0 = y^2 - 1 = (y - 1)(y + 1) \bmod N,$$

i tak $N \mid (y - 1)(y + 1)$. Jednakże $N \mid (y - 1)$ i $N \mid (y + 1)$, ponieważ $y \equiv \pm 1 \bmod N$. Zatem musi być tak, że $\gcd(y - 1, N)$ jest równe jednemu z czynników pierwszych N .

Niech $k = ed - 1$ i zauważmy, że $\varphi(N) \mid k$. Korzystając z Wniosku 8.21, mamy $= 1 \bmod N$ dla $k \times$ wszystkich $x \in \mathbb{Z}_N$. Niech $k = 2ru$ dla u nieparzystej liczby całkowitej; zauważ to

$r = 1$, ponieważ $\varphi(N)$ (a zatem k) jest parzyste. Naszą strategią rozkładu na czynniki N będzie wielokrotne wybieranie jednolitego x_N . Zobacz ciąg

$$x^{t_0}, x^{2u}, \dots, x^{2ru},$$

all modulo N . Każdy wyraz w tym ciągu jest kwadratem poprzedniego wyrazu i , jak właśnie zauważaliśmy, ostatnim wyrazem w ciągu jest 1. Weź największe i (jeśli istnieje), dla którego $y^2 \stackrel{\text{def}}{=} [x]^{2j_m} \pmod N = 1$. Wybierając i , mamy $1 \pmod N$.
 $y^2 = 1$, znaleźliśmy nietrywialny pierwiastek kwadratowy z N i może następnie uwzględnić N , jak omówiono powyżej.

Wszystkie powyższe kroki można wykonać w czasie wielomianowym, więc jedynym pytaniem jest określenie prawdopodobieństwa, ponad wyborem x , że y jest nietrywialnym pierwiastkiem kwadratowym z N . Niech $i \in \{0, \dots, r-1\}$ będzie największą wartością i , dla której istnieje $x \in \mathbb{Z}_N$ nieparzyste, którego istnieje $2iu = 1 \pmod N$. (Ponieważ u jest $(-1)u = -1 =$ takie, że $x \pmod N$, więc definicja nie jest pusta.) Wtedy dla wszystkich $x \in \mathbb{Z}$ mamy $[x]^{2j_i+1} = 1 \pmod N$ i tak $[x]^{2j_m} \pmod N$ jest pierwiastkiem kwadratowym z 1. Zdefiniuj

$$\text{Zły} \stackrel{\text{def}}{=} \{x \mid x^{2j_m} = \pm 1 \pmod N\}$$

i zauważ, że jeśli nasz algorytm wybierze $x \in \text{Bad}$, to znajdzie nietrywialny pierwiastek kwadratowy z 1. Pokazujemy, że Bad jest ścisłą podgrupą \mathbb{Z} , co $\pmod N$; przez Lemat 8.37, oznacza, że $|\text{Bad}| \mid |\mathbb{Z} \pmod N|/2$. Oznacza to, że $x \in \text{Bad}$ (i algorytm znajduje nietrywialny pierwiastek kwadratowy z 1) z prawdopodobieństwem co najmniej $1/2$ w każdej iteracji. Użycie wystarczającej liczby iteracji daje wynik twierdzenia.

Udowodniliśmy teraz, że Bad jest ścisłą podgrupą \mathbb{Z} , a nie $\pmod N$. Najpierw zauważ, że Bad jest pustą, ponieważ $1 \in \text{Bad}$. Co więcej, jeśli $x, x \in \text{Zły}$, to

$$\begin{aligned} 2iu &= 2iu & (xx) \pmod N = \\ (\pm 1) \cdot (\pm 1) &= \pm 1 \pmod N, & = x \end{aligned}$$

i tak $xx \in \text{Bad}$ i Bad jest podgrupą. Aby zobaczyć, że Bad jest ścisłą podgrupą, $2iu = 1 \pmod N$ będzie takie, że $x \in \text{Bad}$ i $x \in \text{Bad}$ (taki x musi istnieć według naszego definicji, niech $x = 1 \pmod N$, to $x \in \text{Bad}$ i gotowe. W przeciwnym razie niech $N = pq$ z p, q liczbą pierwszą i niech $x = (xp, xq)$ będzie chińską reprezentacją x z resztą $2iu$. Ponieważ $x = 1 \pmod N$, wiemy o tym

$$(xp, xq)^{2j_m} = (x_p^{2j_m} x_q^{2j_m}) = (-1, -1) = 1.$$

Ale wtedy $(xp, 1)$ (a raczej element mu odpowiadający) nie jest w Bad , ponieważ

$$(xp, 1)^{2iu} = ([x] \pmod p)^{2iu} = (-1, -1) = \pm 1.$$

To kończy dowód. ■

Zakładając, że faktoring jest trudny, powyższy wynik wyklucza możliwość skutecznego rozwiązania problemu RSA poprzez najpierw obliczenie $d \pmod N$ i e . Nie wyklucza to jednak, że mogą istnieć całkowicie

innego sposobu podejścia do problemu RSA, który nie obejmuje (ani nie implikuje) faktoringu N. Zatem, w oparciu o naszą obecną wiedzę, założenie RSA jest silniejsze niż założenie faktoringu – to znaczy może być tak, że problem RSA można rozwiązać w czasie wielomianowym, chociaż nie można tego zrobić na faktoringu.

Niemniej jednak, gdy GenRSA jest konstruowane w oparciu o GenModulus , jak w Algo-rithm 8.47, przeważa przypuszczenie, że problem RSA jest trudny w porównaniu z GenRSA , ilekroć faktoring jest trudny w porównaniu z GenModulus.

8.3 Założenia kryptograficzne w grupach cyklicznych

W tej sekcji przedstawiamy klasę założeń dotyczących twardości kryptograficznej w grupach cyklicznych. Zaczynamy od ogólnego omówienia grup cyklicznych, po którym następuje abstrakcyjne definiowanie odpowiednich założeń. Następnie przyjrzymy się dwóm konkretnym i powszechnie używanym przykładom grup cyklicznych, w przypadku których uważa się, że te założenia są prawdziwe.

8.3.1 Grupy cykliczne i generatory

Niech G będzie skończoną grupą rzędu m . Dla dowolnego $g \in G$ rozważ zbiór

$$G^{\text{zdecydowanie}} = \{g^0, g^1, \dots\}.$$

(Ostrzegamy czytelnika, że jeśli G jest grupą nieskończoną, g jest definiowane inaczej.)

Na podstawie Twierdzenia 8.14 mamy $g^m = 1$. Niech $i \in \mathbb{Z}$ będzie najmniejszą dodatnią liczbą całkowitą, którego $g^{i \text{ itd.}} = 1$. Następnie powyższa sekwencja powtarza się po i wyrazach (tj. $g^i, g^{i+1}, \dots, g^{i+m-1} = g^0, g^1, \dots, g^{i-1}$ i tak)

$$\{g^0, g^1, \dots, g^{i-1}\}.$$

Widzimy, że g zawiera co najwyżej i elementów. W rzeczywistości zawiera dokładnie i elementów, ponieważ jeśli $g^j = g^k$, to $j - k$ jest mniejsza od i .

zaprzeczając naszemu wyborowi i jako najmniejszej dodatniej liczby całkowitej, dla której $g^i = g^k$.

Nie jest trudno sprawdzić, że g jest podgrupą G dla dowolnego $g \in G$ (patrz ćwiczenie 8.3); nazywamy g podgrupą utworzoną przez g . Jeżeli rząd podgrupy g to i , to i nazywa się rzędem g ; to jest:

DEFINICJA 8.51 Niech G będzie grupą skończoną, a $g \in G$. Rząd g jest najmniejszą dodatnią liczbą całkowitą i z $g^i = g^j$ dla dowolnych $i, j \in \mathbb{Z}$.

Poniżej znajduje się użyteczny odpowiednik Wniosku 8.15 (dowód jest identyczny):

TWIERDZENIE 8.52 Niech G będzie grupą skończoną, a $g \in G$ elementem rzędu i . Wtedy dla dowolnej liczby całkowitej x mamy $g^x = g^{\lfloor x \rfloor}$.

Możemy udowodnić coś mocniejszego:

TWIERDZENIE 8.53 Niech G będzie grupą skońzoną, a $g \in G$ elementem rzędu i . Następnie $G^x = g^y$ wtedy i tylko wtedy, gdy $x \equiv y \pmod{i}$.

DOWÓD Jeśli $x \equiv y \pmod{i}$ to $[x \pmod{i}] = [y \pmod{i}]$ i poprzednie twierdzenie mówi, że

$$G^x = g^{[x \pmod{i}]} = g^{[y \pmod{i}]} = g^y.$$

Aby uzyskać bardziej interesujący kierunek, $x \equiv y \pmod{i}$. Wtedy $1 = g^{x-y} = g^{[x-y \pmod{i}]}$ powiedz g (używając poprzedniego twierdzenia). Ponieważ $[x-y \pmod{i}] < i$, ale i jest najmniejszą dodatnią liczbą całkowitą z $g = 1$, musimy mieć $[x-y \pmod{i}] = 0$. ■

Element tożsamości dowolnej grupy G jest jedynym elementem rzędu 1 i generuje grupę $\{1\}$. Z drugiej strony, jeśli istnieje element $g \in G$ mający rząd m (gdzie m jest rzędem G), to $g^m = G$. W tym przypadku G nazywamy grupą cykliczną i mówimy, że g jest generatorem G . (Grupa cykliczna może mieć wiele generatorów, dlatego nie możemy mówić o generatorze.)

Jeśli g jest generatorem G , to z definicji każdy element $h \in G$ jest równy x dla pewnego $x \in \{0, \dots, m-1\}$, punkt, do którego powrócimy w następnej sekcji.

Różne elementy tej samej grupy G mogą mieć różne porządkie. Możemy, należy jednak nałożyć pewne ograniczenia na to, jakie mogą być te możliwe zamówienia.

TWIERDZENIE 8.54 Niech G będzie skońzoną grupą rzędu m i powiedzmy, że $g \in G$ ma rząd i . Wtedy $ja \mid M$.

DOWÓD Z twierdzenia 8.14 wiemy, że g implikuje, że $m^i = 1 = g^{i^0}$. Twierdzenie 8.53 0 mod i . ■

Następny wniosek ilustruje siłę tego wyniku:

DODATEK 8.55 Jeśli G jest grupą rzędu pierwszego p , to G jest cykliczne. Ponadto wszystkie elementy G , z wyjątkiem tożsamości, są generatorami G .

DOWÓD Zgodnie z Twierdzeniem 8.54 jedynymi możliwymi porządkami elementów w G są 1 i p . Tylko tożsamość ma rząd 1, więc wszystkie inne elementy mają rząd p i generują G . ■

Grupy rzędu pierwszego tworzą jedną klasę grup cyklicznych. Grupa addytywna Z_N dla $N > 1$ daje kolejny przykład grupy cyklicznej (element 1 jest zawsze generatorem). Następne twierdzenie – szczególny przypadek Twierdzenia A.21 –

podaje ważną dodatkową klasę grup cyklicznych; dowód wykracza poza zakres tej książki, ale można go znaleźć w dowolnym standardowym tekście algebry abstrakcyjnej.

TWIERDZENIE 8.56 Jeśli p jest liczbą pierwszą, to \mathbb{Z}_p jest grupą cykliczną rzędu $p - 1$.

Dla $p > 3$ liczba pierwsza Z nie ma rzędu pierwszego, więc powyższe nie wynika z poprzedniego wniosku.

Przykład 8.57

Rozważmy grupę (addytywną) Z_{15} . Jak zauważliśmy, Z_{15} jest cykliczny, a element 1 jest generatorem, ponieważ $15 \cdot 1 = 0 \pmod{15}$ oraz $i \cdot 1 = i = 0 \pmod{15}$ dla dowolnego $0 < i < 15$ (przypomnijmy, że w tej grupie tożsamość wynosi 0).

Z_{15} ma inne generatory. Na przykład $2 = \{0, 2, 4, \dots, 14, 1, 3, \dots, 13\}$ i tak 2 jest także generatorem.

Nie każdy element generuje Z_{15} . Na przykład element 3 ma rząd 5, ponieważ $5 \cdot 3 = 0 \pmod{15}$, więc 3 nie generuje Z_{15} . Podgrupa 3 składa się z 5 elementów $\{0, 3, 6, 9, 12\}$ i oczywiście jest to podgrupa podlegająca dodaniu modulo 15. Element 10 ma rząd 3, ponieważ $3 \cdot 10 = 0 \pmod{15}$, a podgrupa 10 składa się z 3 elementów $\{0, 5, 10\}$. Rzędy podgrup (tj. 5 i 3) dzielą $|Z_{15}| = 15$ zgodnie z wymogami Twierdzenia 8.54.

Przykład 8.58

Rozważmy (multiplikatywną) grupę Z_{15} rzędu $(5-1)(3-1) = 8$. Mamy $2 = \{1, 2, 4, 8\}$, więc rząd 2 wynosi 4. Zgodnie z Twierdzeniem 8.54, 4 dzieli 8.

Przykład 8.59

Rozważmy grupę (addytywną) Z_p rzędu pierwszego p . Wiemy, że ta grupa jest cykliczna, ale Wniosek 8.55 mówi nam więcej: mianowicie każdy element oprócz 0 jest generatorem. Rzeczywiście, dla dowolnego $h \in \{1, \dots, p-1\}$ i liczby całkowitej $i > 0$ mamy $ih = 0 \pmod{p}$ wtedy i tylko wtedy, gdy $p | ih$. Ale wtedy Twierdzenie 8.3 mówi, że albo $p | h$ lub $p | i$. To pierwsze nie może wystąpić (ponieważ $h < p$), a najmniejsza dodatnia liczba całkowita, dla której może wystąpić to drugie, to $i = p$. Pokazaliśmy zatem, że każdy niezerowy element h ma rząd p (i w ten sposób generuje Z_p), zgodnie z wnioskiem 8.55.

Przykład 8.60

Rozważmy (multiplikatywną) grupę Z_7 , która jest cykliczna zgodnie z Twierdzeniem 8.56. Mamy $2 = \{1, 2, 4\}$, więc 2 nie jest generatorem. Jednakże,

$$3 = \{1, 3, 2, 6, 4, 5\} = Z_7,$$

i tak 3 jest generatorem Z_7 .

Poniższy przykład opiera się na materiale z Sekcji 8.1.5.

Przykład 8.61

Niech G będzie grupą cykliczną rzędu n i niech g będzie generatorem G . Wtedy odwzorowanie $f : \mathbb{Z}_n \rightarrow G$ dane przez $f(a) = g^a$ jest izomorfizmem pomiędzy \mathbb{Z}_n i G . Rzeczywiście, dla $a, a \in \mathbb{Z}_n$ mamy

$$f(a + a) = g^{[a+a \text{ mod } n]} = g^{a+a} = g^a \cdot g^a = f(a) \cdot f(a).$$

Bijektywność f można udowodnić, korzystając z faktu, że n jest rzędem g .

Poprzedni przykład pokazuje, że wszystkie grupy cykliczne tego samego rzędu są izomorficzne, a zatem takie same z algebraicznego punktu widzenia. Podkreślamy, że nie jest to prawdą w sensie obliczeniowym, a w szczególności izomorfizm $f : G \rightarrow \mathbb{Z}_n$ (o którym wiemy, że musi istnieć) nie musi być efektywnie obliczalny.

Kwestia ta powinna stać się jaśniejsza po dyskusji w poniższych sekcjach oraz w Rozdziale 9.

8.3.2 Logarytm dyskretny/Założenia Diffiego-Hellmana

Wprowadzimy teraz kilka problemów obliczeniowych, które można zdefiniować dla dowolnej klasy grup cyklicznych. Dyskusję w tej sekcji zachowamy w sposób abstrakcyjny, a konkretne przykłady grup, w których problemy te uważa się za trudne, rozważymy w sekcjach 8.3.3 i 8.3.4.

Niech G oznacza ogólny algorytm generowania grup, działający w czasie wielomianowym. Jest to algorytm, który na wejściu $1n$ podaje opis grupy cyklicznej G , jej rząd q (przy $q = n$) oraz generator $g \in G$. Opis grupy cyklicznej określa sposób reprezentacji elementów grupy jako ciągi bitów; zakładamy, że każdy element grupy jest reprezentowany przez unikalny ciąg bitów. Wymagamy, aby istniały wydajne algorytmy (mianowicie algorytmy działające w wielomianu czasu $w(n)$) do obliczania operacji grupowej w G , a także do testowania, czy dany ciąg bitów reprezentuje element G .

Efektywne obliczenie operacji grupowej implikuje wydajne algorytmy potęgowania w G (patrz Dodatek B.2.3) i próbkowania jednorodnego elementu $h \in G$ (wystarczy wybrać uniform $x \in \mathbb{Z}_q$ i ustawić $h := g^x$).

Jak omówiono na końcu poprzedniej sekcji, chociaż wszystkie grupy cykliczne danego rzędu są izomorficzne, reprezentacja grupy określa złożoność obliczeniową operacji matematycznych w tej grupie. . . , $g^q - 1\}$ jest całym G . Równoważnie, dla każdego $h \in G$

Jeśli G jest grupą cykliczną rzędu q z generatorem g , to $\{g^{0, 1, \dots, q-1}\}$, G istnieje unikalne $x \in \mathbb{Z}_q$ takie, że $h = g^x$. Kiedy podstawową grupę G rozumiemy z kontekstu, nazywamy G^x to x logarymem dyskretnym h w odniesieniu do g i zapisujemy $x = \log g h$.

(W tym przypadku logarytmy nazywane są „dyskretnymi”, ponieważ przyjmują wartości ze skończonego zakresu, w przeciwieństwie do „standardowych” logarytmów z rachunku różniczkowego, których wartości mieszą się w nieskończonym zbiorze liczb rzeczywistych.) Należy zauważać, że jeśli $g = h$ dla dowolnej liczby całkowitej x , wtedy $[x \text{ mod } q] = \log g h$.

Logarytmy dyskretne podlegają wielu tym samym zasadom, co logarytmy „standardowe”.

Na przykład $\log g \equiv 0$ (gdzie 1 to tożsamość G); dla dowolnej liczby całkowitej r mamy $\log g^r \equiv [r \cdot \log g \bmod q]$; i $\log(g_1g_2) = [\log g_1 + \log g_2] \bmod q$.

Problem logarytmu dyskretnego w grupie cyklicznej G z generatorem g polega na obliczeniu $\log h$ dla jednolitego elementu $h \in G$. Rozważmy następujący eksperyment dla algorytmu G generowania grup, algorytmu A i parametru n:

Doświadczenie logarytmu dyskretnego DLogA,G (n):

1. Uruchom G(1n), aby otrzymać (G, q, g), gdzie G jest grupą cykliczną rzędu q (gdzie q = n), a g jest generatorem G.
2. Wybierz mundur h $\in G$.
3. A ma dane G, q, g, h i wyjścia $x \in \mathbb{Z}_q$.
4. Wynik eksperymentu definiuje się jako 1, jeśli $g^x \equiv h \pmod{q}$ w przeciwnym razie.

DEFINICJA 8.62 Mówimy, że problem logarytmu dyskretnego jest trudny w stosunku do G, jeśli dla wszystkich probabilistycznych algorytmów wielomianowych A istnieje funkcja pomijalna taka, że $\Pr[DLogA,G (n) = 1] \leq \text{negl}(n)$.

Założenie logarytmu dyskretnego jest po prostu założeniem, że istnieje G, dla którego problem logarytmu dyskretnego jest trudny. W poniższych dwóch sekcjach omówiono niektóre algorytmy generowania grup kandydatów G, w przypadku których uważa się, że tak jest.

Problemy Diffiego-Hellmana. Tak zwane problemy Diffiego-Hellmana są powiązane z problemem obliczania logarytmów dyskretnych, ale nie są równoważne. Istnieją dwa ważne warianty: obliczeniowy problem Diffiego-Hellmana (CDH) i decyzyjny problem Diffiego-Hellmana (DDH).

Ustal grupę cykliczną G i generator g $\in G$. Dane elementy $h_1, h_2 \in G$, zdefiniuj $DHg(h_1, h_2) = g^{\log h_1 \cdot \log h_2} \pmod{q}$. Oznacza to, że jeśli $h_1 = g^{x_1}$ i $h_2 = g^{x_2}$, to $DHg(h_1, h_2) = g^{x_1 \cdot x_2} \equiv g^{\log h_1 \cdot \log h_2} \pmod{q}$.

$$DHg(h_1, h_2) = g^{x_1 \cdot x_2} \equiv g^{\log h_1 \cdot \log h_2} \pmod{q}.$$

Problem CDH polega na obliczeniu $DHg(h_1, h_2)$ dla jednakowych h_1 i h_2 . Trudność tego problemu można sformalizować za pomocą naturalnego eksperymentu; szczegóły zostawiamy jako ćwiczenie.

Jeśli problem logarytmu dyskretnego w odniesieniu do pewnego G jest łatwy, to problem CDH również jest następujący: mając h_1 i h_2 , najpierw oblicz $x_1 := \log h_1$, a następnie wyrowadź odpowiedź na x_1 . Natomiast nie jest jasne, czy twardeść dyskretnych problem logarytmu G oznacza, że problem CDH jest również trudny.

Z grubsza mówiąc, problem DDH polega na odróżnieniu $DHg(h_1, h_2)$ od jednolitego elementu grupy, gdy h_1, h_2 są jednorodne. Oznacza to, że biorąc pod uwagę jednorodne h_1, h_2 i element trzeciej grupy h, problem polega na tym, aby zdecydować, czy $h = DHg(h_1, h_2)$, czy też h zostało wybrane jednolicie z G. Formalnie:

DEFINICJA 8.63 Mówimy, że problem DDH jest trudny w stosunku do G, jeśli dla wszystkich probabilistycznych algorytmów A w czasie wielomianowym istnieje funkcja pomijalna taka , że

$$\Pr[A(G, q, g, gx, gy, gz) = 1] - \Pr[A(G, q, g, gx, gy, gxy) = 1] = \text{negl}(n),$$

gdzie w każdym przypadku prawdopodobieństwa przejmuje się z eksperymentu, w którym $G(1n)$ wyprowadza (G, q, g) , a następnie wybiera się jednorodne $x, y, z \in \mathbb{Z}_q$. (Zauważ, że gdy z jest jednorodne w \mathbb{Z}_q , to $g^x \cdot g^y \cdot g^z$ jest równomiernie rozłożone w G .)

Widzieliśmy już, że jeśli problem logarytmu dyskretnego jest łatwy w stosunku do pewnego G , to problem CDH również jest łatwy. Podobnie, jeśli problem CDH jest łatwy w stosunku do G , to problem DDH również jest łatwy; zostaniesz poproszony o pokazanie tego w ćwiczeniu 8.15. Nie wydaje się jednak, aby sytuacja odwrotna była prawdziwa i istnieją przykłady grup, w których problemy logarytmu dyskretnego i CDH uważa się za trudne, mimo że problem DDH jest łatwy; patrz ćwiczenie 13.15.

Korzystanie z grup pierwszego rzędu

Istnieją różne (klasy) grup cyklicznych, w przypadku których uważa się, że problemy logarytmu dyskretnego i problemy Diffiego-Hellmana są trudne. Preferowane są jednak grupy cykliczne rzędu pierwszego, z powodów, które teraz wyjaśnimy.

Jednym z powodów preferowania grup rzędu pierwszego jest to, że w pewnym sensie problem logarytmu dyskretnego jest w takich grupach najtrudniejszy. Jest to konsekwencja algorytmu Pohliga-Hellmana opisanego w rozdziale 9, który pokazuje, że problem logarytmu dyskretnego w grupie rzędu q staje się łatwiejszy, jeśli q ma (małe) czynniki pierwsze. Nie musi to koniecznie oznaczać, że problem logarytmu dyskretnego jest łatwy w grupach rzędu innego niż pierwszy; oznacza to jedynie, że problem staje się łatwiejszy.

Z powyższym wiąże się fakt, że problem DDH jest łatwy, jeśli rząd grupowy q ma małe czynniki pierwsze. Jako przykład tego zjawiska odsyłamy do ćwiczenia 13.15.

Drugą motywacją do korzystania z grup rzędu pierwszego jest to, że znalezienie generatora w takich grupach jest trywialne. Wynika to z Wniosku 8.55, który mówi, że każdy element grupy pierwszego rzędu (z wyjątkiem tożsamości) jest generatorem. Natomiast skuteczne znalezienie generatora dowolnej grupy cyklicznej wymaga znajomości rozkładu na czynniki rzędu grup (patrz Dodatek B.3).

Dowody bezpieczeństwa niektórych konstrukcji kryptograficznych wymagają obliczenia odwrotności multiplikatywnych niektórych wykładników (przykład zobaczymy w podrozdziale 8.4.2). Gdy porządek grupowy jest liczbą pierwszą, każdy niezerowy wykładnik będzie odwracalny, co umożliwi wykonanie tego obliczenia.

Ostatni powód pracy z grupami rzędu pierwszego ma zastosowanie w sytuacjach, gdy decyzyjny problem Diffiego – Hellmana powinien być trudny. Ustalając grupę G za pomocą generatora g , problem DDH sprowadza się do rozróżnienia

krotki postaci $(h_1, h_2, DHg(h_1, h_2))$ dla jednorodnych h_1, h_2 i krotki postaci (h_1, h_2, y) , dla jednorodnych h_1, h_2, y . Warunkiem koniecznym, aby problem DDH był trudny, jest to, że $DHg(h_1, h_2)$ samo w sobie powinno być nie do odróżnienia od jednolitego elementu grupowego. Można wykazać, że $DHg(h_1, h_2)$ jest „bliskie” jednorodności (w pewnym sensie, którego tutaj nie definiujemy), gdy porządek grupowy q jest liczbą pierwszą, co w przeciwnym razie nie jest prawdą.

8.3.3 Praca w (podgrupach) Z_p

P

Grupy postaci Z_p , dla p prime podaj jedną klasę grup cyklicznych, dla których uważa się, że problem logarytmu dyskretnego jest trudny. Konkretnie, niech G będzie algorytmem, który na wejściu $1n$ wybiera jednolitą n -bitową liczbę pierwszą p i wyprowadza p . oraz rząd grupowy $q = p - 1$ wraz z generatorem g Z omawia efektywne Z_p (Sekcja 8.2.1) algorytmy wyboru losowej liczby pierwszej, a Dodatek B.3 biorąc pod uwagę faktoryzację p - generator Z . Reprezentacja Z tutaj jest trywialny, w Z_p (1.) pokazuje, jak efektywnie znaleźć którym elementy są reprezentowane jako liczby całkowite od 1 do $p - 1$. Przypuszcza się, że problem logarytmu dyskretnego jest trudny w porównaniu z G tego rodzaju.

Grupa cykliczna Z_p (dla $p > 3$ liczba pierwsza) nie ma jednak rzędu pierwszego. (Preferencje dla grup rzędu pierwszego omówiono w poprzedniej sekcji). Bardziej problematyczny jest fakt, że decyzyjny problem Diffiego-Hellmana nie jest na ogół trudny w takich grupach (patrz ćwiczenie 13.15) i dlatego są one niedopuszczalne dla zastosowań kryptograficznych opartych na założeniu DDH, które omówimy w późniejszych rozdziałach.

Problemy te można rozwiązać za pomocą podgrupy rzędu pierwszego Z_p . Pozwalać $p = rq + 1$ gdzie zarówno p , jak i q są liczbami pierwszymi. Udowodnimy, że Z_p ma podgrupę G rzędu q daną przez zbiór r -tych reszt modulo p , tj. zbiór elementów $\{[h \cdot r \text{ mod } p] \mid h \in Z_p\}$ które są równe r -tej potędze pewnego $h \in Z_p$.

TWIERDZENIE 8.64 Niech $p = rq + 1$ z p, q liczbą pierwszą. Następnie

$$G \stackrel{\text{def}}{=} \{[h \cdot r \text{ mod } p] \mid h \in Z_p\}$$

jest podgrupą Z_p zamówienia q .

DOWÓD Dowód, że G jest podgrupą jest prosty i zostanie pominięty.

Dowodzimy, że G ma rząd q , pokazując, że funkcja $f_r : Z_p \rightarrow G$ określona przez $f_r(g) = [g \cdot r \text{ mod } p]$ jest funkcją r do 1. (Ponieważ $|Z_p| = p$ i $|G| = (p - 1)/r = q$) Aby to $p \mid r = p - 1$, to zobaczyć, niech g będzie generatorem Z_p , $g^p \equiv 1 \pmod{p}$. Wtedy $[g^i \cdot r \text{ mod } p] = g^{ir} \equiv g^0 \pmod{p}$. Zgodnie z Twierdzeniem 8.53 mamy tak, że $G = \{g^i \cdot r \mid i = 0, 1, \dots, p-1\}$.

wtedy i tylko wtedy, gdy $ir \equiv jr \pmod{p-1}$ lub, równoważnie, $p-1 \mid (i-j)r$.

Ponieważ $p-1 = rq$, jest to równoważne $q \mid (i-j)r$. Dla dowolnego ustalonego $j \in \{0, 1, \dots, p-2\}$, oznacza to, że zbiór wartości $i \in \{0, 1, \dots, p-2\}$ dla którego $g^i \cdot r \equiv g^j \cdot r$

dokładnie zbiór r różnych wartości

$$\{j, j+q, j+2q, \dots, j+(r-1)q\},$$

wszystko zredukowane modulo $p - 1$. (Zauważ, że $j + rq = j \bmod (p - 1)$.) Dowodzi to, że f_r jest funkcją r do -1. ■

Oprócz pokazania istnienia odpowiedniej podgrupy, z twierdzenia wynika również, że łatwo jest wygenerować jednorodny element G i sprawdzić, czy dany element Z leży w G . W szczególności wybierając jednolitego elementu G można dokonać poprzez wybór jednorodnego $h \in Z$ i obliczanie $[h \ bmod p]$. Ponieważ G ma porządek pierwszy, każdy element w G , z wyjątkiem tożsamości, jest generatorem G .

Wreszcie można określić, czy dowolne $h \in Z$ czy $h \ bmod p = 1$ jest również w G , sprawdzając $h^p \bmod p$. Aby zobaczyć, że to działa, niech $h = g^i$ dla generatora gazowego $Z \bmod p$ i ja $\{0, \dots, p-2\}$. Następnie

$$\begin{aligned} \text{godz. } q = 1 \bmod p \quad \text{sol } h^{iq} &= 1 \bmod p \\ iq = 0 \bmod (p-1) \quad rq \mid q &\quad r \mid i, \end{aligned}$$

używając Propozycji 8.53. Zatem $h = g^i = g^{kr} = (g \ bmod p)^r$ dla pewnego $c \in \mathbb{Z}$ i $h \in G$.

Algorytm 8.65 podsumowuje powyższą dyskusję. W algorytmie niech n oznacza długość q , rzad grupy, i niech oznacza długość p , używany moduł. Zależność pomiędzy tymi parametrami omówiono poniżej.

Wybieranie. Niech $n = q \cdot i = p$. Znane są dwa typy algorytmów do obliczania logarytmów dyskretnych w podgrupach rzędu q z tą samą i : te, które działają w czasie $O(\sqrt{p})$ (patrz rozdział 9.2), i te, które działają w czasie $O(2\sqrt{p})$ i te, które działają w czasie $2O(\sqrt{\log p})$. Ustalając pożądany parametr bezpieczeństwa n , parametr należy dobrze tak, aby zrównoważyć te czasy. (Jeśli jest mniejszy, bezpieczeństwo jest zmniejszone; jeśli jest większe, operacje w G będą mniej wydajne bez żadnego zwiększenia bezpieczeństwa.) Zobacz także Sekcję 9.3.

ALGORYTM 8.65 Algorytm generowania grup G

```
Wejście: Parametr bezpieczeństwa  $n$ , parametr  $(n)$   

Wynik: Grupa cykliczna  $G$ , jej (pierwszy) rzad  $q$  i generator  $g$  generująca jednolitą  $n$ -bitową liczbę pierwszą  $q$   

generująca -bitową liczbę pierwszą  $p$  taką, że  $q \mid (p-1)$  // pomijamy szczegóły tego, jak to jest  

wybierz mundur  $h \in \mathbb{Z}_p$  zrobione przy  $h = 1$   

set  $g := [h(p-1)/q \bmod p]$   

return  $p, q, g$  //  $G$  jest podgrupą rzędu  $q$  zbioru  $\mathbb{Z}_p$ 
```

W praktyce standaryzowane wartości (np. zalecane przez NIST) dla p , q i a generator g i nie ma potrzeby generowania własnych parametrów.

Przykład 8.66

Rozważmy grupę Z_{11} rzędu 10. Spróbujmy znaleźć generator tego Grupa. Rozważ wypróbowanie 2:

Potęgi liczby 2: 2	0	2 ¹	2 ²	2 ³	2 ⁴	2 ⁵	2 ⁶	2 ⁷	2 ⁸	2 ⁹
Wartości: 1 2 4 8 5 10 9 7 3 6										

(Wszystkie powyższe wartości są obliczane modulo 11.) Za pierwszym razem mieliśmy szczęście – numer 2 to generator! Spróbujmy 3:

Potęgi 3: 3	0	3 ¹	3 ²	3 ³	3 ⁴	3 ⁵	3 ⁶	3 ⁷	3 ⁸	3 ⁹
Wartości: 1 3 9 5 4 1 3 9 5 4										

Widzimy, że 3 nie jest generatorem całej grupy. Raczej generuje podgrupę $G = \{1, 3, 4, 5, 9\}$ rzędu 5. Zobaczmy teraz, co stanie się z 10:

Potęgi 10: 10	0	10 ¹	10 ²	10 ³	10 ⁴	10 ⁵	10 ⁶	10 ⁷	10 ⁸	10 ⁹
Wartości: 1		10	10	10	10	10	10	10	10	10

W tym przypadku generujemy podgrupę rzędu 2.

Dla celów kryptograficznych chcemy pracować w grupie pierwszego rzędu. Od $11 = 2 \cdot 5 + 1$ możemy zastosować Twierdzenie 8.64 przy $q = 5$ i $r = 2$ lub przy $q = 2$ i $r = 5$. W pierwszym przypadku twierdzenie mówi nam, że kwadraty wszystkich elementów Z_{11} powinno dać podgrupę rzędu 5. Można to łatwo zweryfikować:

Element: 1 2 3 4 5 6 7 8 9 10
Kwadrat: 1 4 9 5 3 3 5 9 4 1

Widzieliśmy powyżej, że 3 jest generatorem tej podgrupy. (Właściwie od podgrupy jest liczbą pierwszą, każdy element podgrupy oprócz 1 jest generatorem podgrupy.) Biorąc $q = 2$ i $r = 5$, Twierdzenie 8.64 mówi nam, że biorąc Piąta potęga da podgrupę rzędu 2. Można sprawdzić, że to daje podgrupę rzędu 2 wygenerowaną przez 10, z którą spotkaliśmy się wcześniej.

Podgrupy ciał skończonych. Uważa się również, że problem logarytmu dyskretnego jest być trudne w grupie multiplikatywnej skończonego ciała o dużej charakterystyce gdy używana jest reprezentacja wielomianowa. (Załącznik A.5 zawiera streszczenie tło na ciałach skończonych.) Przypomnijmy, że dla dowolnej liczby pierwszej p i liczby całkowitej k istnieje (unikalne) pole F_{pk} rzędu p^k ; grupa multiplikatywna F_{pk} tego pole jest cykliczną grupą rzędu $p^k - 1$ (por. Twierdzenie A.21). Jeśli q jest dużą liczbą pierwszą wspólnym $p^k - 1$, wówczas Twierdzenie 8.64 pokazuje, że F_{pk} ma cykliczną podgrupę p^k zamówionej q . (Jedyna własność Z_{p^k} użyciśmy w dowodzie tego twierdzenia, że Z_{p^k} jest cykliczny.) Daje to inny wybór grup pierwego rzędu, w których Uważa się, że problemy logarytmu dyskretnego i problemów Diffiego-Hellmana są trudne. Nasze leczenie Z_{p^k} w tej sekcji odpowiada specjalnemu przypadkowi $k = 1$.

8.3.4 Krzywe eliptyczne

Wszystkie grupy, na których się dotychczas skupialiśmy, opierały się bezpośrednio na arytmetyce modułowej. Inną klasę grup ważnych dla kryptografii stanowią grupy składające się z punktów na krzywych eliptycznych. Takie grupy są szczególnie interesujące z kryptograficznego punktu widzenia, ponieważ w przeciwieństwie do Z lub grupy multiplikatywnej ciała skońzonego, obecnie nie są znane żadne algorytmy czasu p podwykładniczego do rozwiązywania problemu logarytmu dyskretnego w grupach krzywych eliptycznych, jeśli zostaną odpowiednio wybrane. (Dalsze omówienie można znaleźć w sekcji 9.3.) W przypadku kryptosystemów opartych na logarytmie dyskretnym lub założeniach Diffiego-Hellmana oznacza to, że implementacje oparte na grupach krzywych eliptycznych będą bardziej wydajne niż implementacje oparte na podgrupach rzędu pierwszego Z na dowolnym poziomie bezpieczeństwa. W tej części przedstawiamy jedynie krótkie wprowadzenie do tego obszaru. Głębsze zrozumienie omawianych tu zagadnień wymaga bardziej wyrafinowanej matematyki, niż jesteśmy skłonni założyć ze strony czytelnika. Osobom zainteresowanym dalszym zgłębianiem tego tematu zaleca się zapoznanie się z literaturą znajdującą się na końcu tego rozdziału.

Niech $p = 5$ będzie liczbą pierwszą. Rozważmy równanie E ze zmiennymi x i y postaci:

$$\text{lata}^2 - 3 = x^3 + \text{Topór} + B \bmod p, \quad (8.1)$$

gdzie $A, B \in \mathbb{Z}_p$ są stałymi przy $4A^3 + 27B^2 = 0 \bmod p$. (To gwarantuje, że $Ax + B = 0$ równanie x oznacza $x^3 \bmod p$ nie ma powtarzających się pierwiastków.) Niech $E(\mathbb{Z}_p)$ zbiór par $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$ spełniający powyższe równanie wraz ze specjalną wartością O którego cel omówimy wkrótce; to jest,

$$E(\mathbb{Z}_p) \stackrel{\text{def}}{=} \{(x, y) \mid x, y \in \mathbb{Z}_p \text{ i } y^2 - 3 = x^3 + \text{Topór} + B \bmod p\} \cup \{O\}.$$

Elementy $E(\mathbb{Z}_p)$ nazywane są punktami na krzywej eliptycznej E określonej równaniem (8.1), a O nazywa się punktem w nieskończoności.

Przykład 8.67

Element $y \in \mathbb{Z}$, który jest resztą kwadratową modulo p , jeśli istnieje $x \in \mathbb{Z}_{p-1}$ taki $x^2 \equiv y \pmod p$; w takim przypadku mówimy, że x jest pierwiastkiem kwadratowym z y . Dla $p > 2$ połowa elementów \mathbb{Z} to reszty kwadratowe, a każda reszta kwadratowa ma dokładnie dwa pierwiastki kwadratowe. (Patrz rozdział 13.4.1.)

Niech $f(x) = x^3 + 3x + 3$ i rozważ krzywą E: $y^2 = f(x) \bmod 7$. Każda wartość x , dla której $f(x)$ jest resztą kwadratową modulo 7, daje dwa punkty na krzywej; wartości x , dla których $f(x)$ jest resztą niekwadratową, nie są włączone

³Teorię można dostosować do przypadku $p = 2$ lub 3 , ale wprowadza to dodatkowe komplikacje. Krzywe eliptyczne można w rzeczywistości zdefiniować na dowolnych, skończonych lub nieskończonych polach (por. sekcja A.5), a nasza dyskusja w dużej mierze sprowadza się do pól charakterystycznych, które nie są równe 2 lub 3. Krzywe binarne (tj. krzywe na polach o cecha 2) są szczególnie ważne w kryptografii, ale nie będziemy ich tutaj omawiać.

krzywa; wartości x dla których $f(x) = 0 \pmod{7}$ dają jeden punkt na krzywej.
Dzięki temu możemy wyznaczyć punkty na krzywej:

- $f(0) = 3 \pmod{7}$, kwadratowy nieresztowy modulo 7.
- $f(1) = 0 \pmod{7}$, więc otrzymujemy punkt $(1, 0) \in E(\mathbb{Z}_7)$.
- $f(2) = 3 \pmod{7}$, kwadratowy nieresztowy modulo 7.
- $f(3) = 4 \pmod{7}$, reszta kwadratowa modulo 7 z pierwiastkami kwadratowymi 2 i 5.
Daje to punkty $(3, 2), (3, 5) \in E(\mathbb{Z}_7)$.
- $f(4) = 2 \pmod{7}$, reszta kwadratowa modulo 7 z pierwiastkami kwadratowymi 3 i 4.
Daje to punkty $(4, 3), (4, 4) \in E(\mathbb{Z}_7)$.
- $f(5) = 3 \pmod{7}$, kwadratowy nieresztowy modulo 7.
- $f(6) = 6 \pmod{7}$, kwadratowy nieresztowy modulo 7.

Łącznie z punktem w nieskończoności, w $E(\mathbb{Z}_7)$ jest 6 punktów.

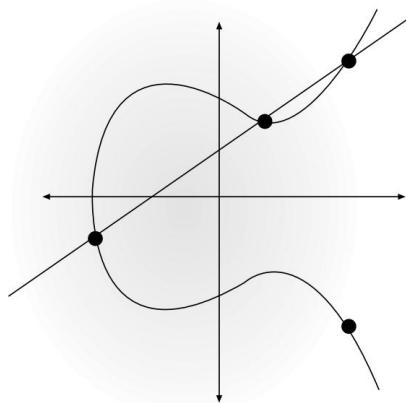
Przydatnym sposobem myślenia o $E(\mathbb{Z}_p)$ jest spojrzenie na wykres równania (8.1) w funkcji liczb rzeczywistych (tj. równanie $y = x^3 + Ax + B$ bez redukcji modulo p), jak na rysunku 8.2. Liczba ta nie odpowiada dokładnie $E(\mathbb{Z}_p)$, bo np. $E(\mathbb{Z}_p)$ ma skońzoną liczbę punktów (\mathbb{Z}_p jest przecież skończonym zbiorem), natomiast istnieje nieskończona liczba rozwiązań tego samego równania, jeśli pozwalamy, aby x i y obejmowały wszystkie liczby rzeczywiste. Niemniej jednak obraz dostarcza użytecznej intuicji. Na takiej figurze można pomyśleć o „punkcie w nieskończoności” O jako znajdującym się na szczytce osi Y i leżącym na każdej linii pionowej.

Można wykazać, że każda prosta przecinająca $E(\mathbb{Z}_p)$ przecina ją dokładnie w 3 punktach, przy czym (1) punkt P jest liczony dwukrotnie, jeśli prosta jest styczna do krzywej w punkcie P oraz (2) punkt w nieskończoności jest również liczone, gdy linia jest pionowa.
Fakt ten służy do zdefiniowania operacji binarnej zwanej „dodawaniem” i oznaczonej znakiem $+$ na punktach $E(\mathbb{Z}_p)$ w następujący sposób:

- Punkt O jest zdefiniowany jako (addytywna) tożsamość; to znaczy dla wszystkich $P \in E(\mathbb{Z}_p)$ definiujemy $P + O = O + P = P$.
- Dla dwóch punktów $P_1, P_2 = O$ na E obliczamy ich sumę $P_1 + P_2$ przeciągając linię przez P_1, P_2 (jeśli $P_1 = P_2$ to narysuj linię styczną do krzywej w P_1) i znajdując trzeci punkt przecięcia P_3 tej linii z $E(\mathbb{Z}_p)$; trzeci punkt przecięcia może wynosić $P_3 = O$, jeśli linia jest pionowa. Jeśli $P_3 = (x, y) = O$, to definiujemy $P_1 + P_2 = (x, -y)$.
def

(Graficznie odpowiada to odzwierciedleniu P_3 na osi x .) Jeśli $P_3 = O$, to $P_1 + P_2 = O$.

Jeżeli $P = (x, y) = O$ jest punktem $E(\mathbb{Z}_p)$, to $P = (x, -y)$ (co jest oczywiście także punktem $E(\mathbb{Z}_p)$) jest jedyną odwrotnością P . Rzeczywiście, linia



RYSUNEK 8.2: Krzywa eliptyczna nad liczbami rzeczywistymi.

przez (x, y) i $(x, -y)$ jest pionowe, więc z reguły dodawania wynika, że $P + (-P) = O$. (Jeśli $y = 0$, to $P = (x, y) = (x, -y) = -P$, ale wtedy styczna w punkcie P będzie pionowa, więc tutaj również $P + (-P) = O$). Oczywiście, $O = O$.

Konkretnie opracowanie prawa dodawania jest proste, ale żmudne.

Niech $P_1 = (x_1, y_1)$ i $P_2 = (x_2, y_2)$ będą dwoma punktami w $E(\mathbb{Z}_p)$, gdzie $P_1, P_2 \neq O$ i E jak w równaniu (8.1). Dla uproszczenia założymy, że $x_1 = x_2$ (zajmowanie się przypadkiem $x_1 = x_2$ jest nadal proste, ale jeszcze bardziej nudzące). Nachylenie linii przechodzącej przez te punkty wynosi

$$\text{zdecydowanie } m = \frac{y_2 - y_1}{x_2 - x_1} \pmod{p};$$

nasze założenie, że $x_1 = x_2$ oznacza, że istnieje odwrotność $(x_2 - x_1)$ modulo p . Linia przechodząca przez P_1 i P_2 ma równanie

$$y = m \cdot (x - x_1) + y_1 \pmod{p}. \quad (8.2)$$

Aby znaleźć trzeci punkt przecięcia tej prostej z E , podstawiamy powyższe do równania, aby otrzymać E

$$m \cdot (x - x_1) + y_1 \stackrel{2}{=} x^3 + Topór + B \pmod{p}$$

Wartości x spełniające to równanie to x_1, x_2 i

$$x_3 \stackrel{\text{def}}{=} [m^2 - x_1 - x_2] \pmod{p}.$$

Pierwsze dwa rozwiązania odpowiadają pierwotnym punktom P_1 i P_2 , natomiast trzecie jest współrzędną x trzeciego punktu przecięcia P_3 . Podstawiając x_3 do równania (8.2) stwierdzamy, że współrzędna y odpowiadająca x_3 wynosi

$y3 = [m \cdot (x3 - x1) + y1 \bmod p]$. Aby uzyskać pożądaną odpowiedź $P1 + P2$, odwracamy znak y , aby otrzymać:

$$(x1, y1) + (x2, y2) = [m2 - x1 - x2 \bmod p], [m \cdot (x1 - x3) - y1 \bmod p]$$

Podsumowujemy i rozszerzamy to w następującym twierdzeniu.

TWIERDZENIE 8.68 Niech $p = 5$ będzie liczbą pierwszą, a E będzie krzywą eliptyczną $= x^2 + 3Ax + B \bmod p$ gdzie $4A^3 + 27B^2 = 0 \bmod p$. Niech $P1, P2 = O$ dane przez y będą na E , gdzie $P1 = (x1, y1)$ i $P2 = (x2, y2)$.

1. Jeśli $x1 = x2$, to $P1 + P2 = (x3, y3)$ z

$$x3 = [m2 - x1 - x2 \bmod p] \text{ i } y3 = [m \cdot (x1 - x3) - y1 \bmod p],$$

$$\text{gdzie } m = \frac{y2-y1}{x2-x1} \bmod \text{str.}$$

2. Jeśli $x1 = x2$, ale $y1 = y2$, to $P1 = P2$ i tak $P1 + P2 = O$.

3. Jeśli $P1 = P2$ i $y1 = 0$, to $P1 + P2 = 2P1 = O$.

4. Jeżeli $P1 = P2$ i $y1 = 0$ to $P1 + P2 = 2P1 = (x3, y3)$ przy

$$x3 = [m2 - 2x1 \bmod p] \text{ i } y3 = [m \cdot (x1 - x3) - y1 \bmod p],$$

$$\text{gdzie } m = \frac{2x_1 + A}{lata} \bmod \text{str. .2}$$

Nieco zdumiewające jest to, że zbiór punktów $E(Zp)$ wraz z zdefiniowaną powyżej zasadą dodawania tworzy grupę abelową, zwaną grupą o krzywej eliptycznej E . Przemienność wynika ze sposobu zdefiniowania dodawania, O pełni rolę tożsamości i widzieliśmy już, że każdy punkt na $E(Zp)$ ma odwrotność w $E(Zp)$. Właściwością trudną do sprawdzenia jest łączność, którą niedowierzający czytelnik może sprawdzić poprzez żmudne obliczenia. Bardziej pouczający dowód, który nie wymaga wyraźnych obliczeń, opiera się na geometrii algebraicznej.

Przykład 8.69

Rozważmy krzywą z Przykładu 8.67. Pokazujemy łączność dla trzech konkretnych punktów. Niech $P1 = (1, 0)$, $P2 = Q2 = (4, 3)$. Obliczając $P1 + P2$ otrzymujemy $m = [(3 - 0) \cdot (4 - 1) - 1 \bmod 7] = 1$ i $[12 - 1 - 4 \bmod 7] = 3$. Zatem

$$P3 \stackrel{\text{def}}{=} P1 + P2 = (3, [1 \cdot (1 - 3) - 0 \bmod 7]) = (3, 5);$$

zauważ, że rzeczywiście jest to punkt na $E(Z7)$. Jeśli następnie obliczymy $P3 + Q2$, otrzymamy $m = [(3 - 5) \cdot (4 - 3) - 1 \bmod 7] = 5$ i $[52 - 3 - 4 \bmod 7] = 4$. Zatem

$$(P1 + P2) + Q2 = P3 + Q2 = (4, [5 \cdot (3 - 4) - 5 \bmod 7]) = (4, 4).$$

Jeżeli obliczymy $P_2 + Q_2 = 2P_2$ otrzymamy $m = [(3 \cdot 4 + 2 + 3) \cdot (2 \cdot 3) - 1 \bmod 7] = 5$
 i $[52 - 2 \cdot 4 \bmod 7] = 3$. Zatem

$$P_3 \stackrel{\text{def}}{=} P_2 + Q_2 = (3, [5 \cdot (4 - 3) - 3 \bmod 7]) = (3, 2).$$

Jeśli następnie obliczymy $P_1 + P_3$ znajdujemy $m = [2 \cdot (3 - 1) - 1 \bmod 7] = 1$ i
 $[12 - 1 - 3 \bmod 7] = 4$. Zatem

$$P_1 + (P_2 + Q_2) = P_1 + P_3 = (4, [1 \cdot (1 - 4) - 0 \bmod 7]) = (4, 4),$$

i $P_1 + (P_2 + Q_2) = (P_1 + P_2) + Q_2$.

Przypomnijmy, że gdy grupa jest zapisywana addytywnie, „potęgowanie” odpowiada wielokrotnemu dodawaniu. Zatem, jeśli ustalimy jakiś punkt P w grupie krzywych eliptycznych, problem logarytmu dyskretnego staje się (nieformalnie) problemem obliczenia liczby całkowitej x z xP , podczas gdy decyzyjny problem Diffiego-Hellmana staje się (nieformalnie) problemem odróżnianie krotek postaci (aP, bP, abP) od krotek postaci (aP, bP, cP) . Uważa się, że problemy te są trudne w grupach krzywych eliptycznych (lub ich podgrupach) dużego rzędu pierwszego, z zastrzeżeniem kilku warunków technicznych, o których wspomnijmy przelotnie poniżej.

Jeśli chcemy grupy (lub podgrupy) krzywych eliptycznych dużego rzędu pierwszego, pierwszym pytaniem, na które musimy odpowiedzieć, jest: jak duże są grupy krzywych eliptycznych? Jak zauważono w przykładzie 8.67², równanie $y = f(x) \bmod p$ ma dwa rozwiązania, gdy $f(x)$ jest resztą kwadratową, i jedno rozwiązanie, gdy $f(x) = 0$. Ponieważ połowa elementów w Z to reszty kwadratowe, heurystycznie spodziewamy się znaleźć $p - 2 \cdot (p - 1)/2 + 1 = p$ punktów na krzywej. Uwzględniając punkt w nieskończoności, oznacza to, że w grupie krzywej eliptycznej nad Z_p powinno znajdować się około $p + 1$ punktów. Granica Hassego mówi, że to oszacowanie heurystyczne jest dokładne w tym sensie, że każda grupa krzywych eliptycznych ma „prawie” tyle punktów.

TWIERDZENIE 8.70 (Hasse związany) Niech p będzie liczbą pierwszą i niech E będzie krzywą eliptyczną nad Z_p . Wtedy $p + 1 - |E(Z_p)| = p + 1 + 2 \bmod p$.

To ograniczenie implikuje, że zawsze łatwo jest znaleźć punkt na danej elipsie. $2 = f(x) \bmod p$, czy $2 \bmod p$: po prostu wybierz jednorodny $x \in Z_p$, sprawdź, czy krzywa $f(x)$ y wynosi resztę kwadratową i – jeśli więc – niech y będzie pierwiastkiem kwadratowym z $f(x)$. (Algorytmy wyznaczania resztości kwadratowej i obliczania pierwiastków kwadratowych modulo a prime omówiono w rozdziale 13.) Ponieważ punktów na krzywej eliptycznej jest mnóstwo, nie będziemy musieli wypróbowywać zbyt wielu wartości x, zanim znajdziemy punkt.

Ograniczenie Hassego daje jedynie zakres rozmiaru grupy o krzywej eliptycznej. Jednakże dla ustalonej liczby pierwszej p rzad losowej krzywej eliptycznej nad Z_p (mianowicie krzywej określonej równaniem (8.1), w której A, B są wybrane jednostajnie w Z_p z zastrzeżeniem ograniczenia $4A^3 + 27B^2 = 0 \bmod p$) jest heurystycznie uznane za „bliskie” rozkładowi równomiernie w przedziale Hassego. Istnieją również wydajne algorytmy – których opis i analiza znacznie wykraczają poza zakres tej książki – służące do liczenia liczby punktów na elipsie

krzywa. Sugeruje to podejście do generowania parametrów krzywej eliptycznej, jak w algorytmie 8.71.

ALGORYTM 8.71 Algorytm generowania grup krzywych eliptycznych G

Wejście: Parametr bezpieczeństwa 1n

Wynik: Grupa cykliczna G, jej (pierwszy) rząd q i generator g generują jednolitą

n-bitową liczbę pierwszą p, dopóki q nie

będzie n-bitową liczbą pierwszą do:

wybierz A, B $\in \mathbb{Z}_p$ gdzie $4A = 0 \pmod{p}$, tworząc krzywą

eliptyczną E jak w równaniu (8.1)

niech q będzie liczbą punktów na E(\mathbb{Z}_p) wybierz g

$E(\mathbb{Z}_p) \setminus \{O\}$ return (A, B, p),

q, g

// G jest grupą o krzywej eliptycznej E

Pewne klasy krzywych są uważane za słabe kryptograficznie i należy ich unikać. Należą do nich grupy krzywych eliptycznych nad \mathbb{Z}_p , których rząd jest równy p (krzywe anomalne) lub $p+1$ (krzywe nadzosobne) lub których rząd dzieli się na -1 dla „małych” k. Pełne omówienie wykracza k s poza zakres tej książki. W praktyce stosuje się znormalizowane krzywe (takie jak te zalecane przez NIST) i nie zaleca się tworzenia własnych krzywych.

Względy wydajności

Zakończymy tę sekcję bardzo krótkim omówieniem pewnego standardu poprawę wydajności przy stosowaniu krzywych eliptycznych.

Kompresja punktowa. Przydatną obserwacją jest to, że liczbę bitów potrzebnych do przedstawienia punktu na krzywej eliptycznej można zmniejszyć prawie o połowę. Aby to zobaczyć, zauważ, że dla dowolnego punktu (x, y) krzywej eliptycznej $E : y^2 = f(x) \pmod{p}$ istnieją co najwyżej dwa punkty na krzywej, które mają wspólnązę x : mianowicie (x, y) i $(x, -y)$. (Możliwe, że $y = 0$ i w tym przypadku są to te same punkty.) Zatem możemy określić dowolny punkt $P = (x, y)$ na podstawie jego współrzędnej x i bitu b, który rozróżnia (co najwyżej) dwa możliwości wartości jego współrzędnej y . Jednym z wygodnych sposobów na osiągnięcie tego jest ustawienie $b = 0$, jeśli $y < p/2$ i $b = 1$ w przeciwnym razie. Biorąc pod uwagę x i b , możemy odzyskać P , obliczając dwa $y_1 = f(x) \pmod{p}$; ponieważ $y_1 = y_2 \pmod{p}$ pierwiastki kwadratowe y_1, y_2 z równania y i tak $y_1 = p - y_2$, dokładnie jeden z y_1, y_2 będzie mniejszy niż $p/2$.

Współrzędne rzutowe. Reprezentowanie punktów na krzywej eliptycznej w sposób, w jaki robiliśmy to dotychczas – w którym punkt P na krzywej eliptycznej jest opisany przez parę elementów pola (x, y) – nazywa się za pomocą współrzędnych afiniycznych. Istnieją alternatywne sposoby reprezentowania punktów przy użyciu współrzędnych rzutowych, które mogą zapewnić wydajność

ulepszenia. Chociaż te alternatywne reprezentacje można motywować matematycznie, traktujemy je po prostu jako użyteczną pomoc obliczeniową.

Punkty we współrzędnych rzutowych są reprezentowane za pomocą trzech elementów Z_p . Ciekawą cechą jest to, że punkt ma wiele reprezentacji. W przypadku stosowania standardowych współrzędnych rzutowych punkt $P = O$ z reprezentacją (x, y) we współrzędnych afinicznych jest reprezentowany przez dowolną krotkę $(X, Y, Z) \in Z^3$, dla której $X/Z = x \pmod{p}$ i $Y/Z = y \pmod{p}$. Punkt O jest reprezentowany przez dowolną krotkę $(0, Y, 0)$ o $Y = 0$ i $1 \leq Y \leq p-1$. Jedynie punkty $(X, Y, Z) \in Z^3$ dla których $Z = 0$ nie mają reprezentacji we współrzędnych rzutowych. Możemy łatwo dokonywać translacji pomiędzy układami współrzędnych: (x, y) we współrzędnych afinicznych można odwzorować na $(x, y, 1)$ we współrzędnych rzutowych, a (X, Y, Z) (przy $Z = 0$) we współrzędnych rzutowych można odwzorować na reprezentację $([X/Z] \pmod{p}, [Y/Z] \pmod{p})$ we współrzędnych afinicznych.

Główna zaletą stosowania współrzędnych rzutowych jest to, że możemy dodawać punkty bez konieczności obliczania odwrotności modulo p . (Dodawanie punktów do współrzędnych afinicznych wymaga obliczenia odwrotności; patrz Twierdzenie 8.68.) Osiągamy to poprzez wykorzystanie faktu, że punkty mają wielokrotną reprezentację. Aby to zobaczyć, wypracujmy prawo dodawania dla dwóch punktów $P_1 = (X_1, Y_1, Z_1)$ i $P_2 = (X_2, Y_2, Z_2)$ gdzie $P_1, P_2 \neq O$ (więc $Z_1, Z_2 \neq 0$) i $P_1 = \pm P_2$ (więc $X_1/Z_1 = X_2/Z_2 \pmod{p}$). (Jeśli P_1 lub P_2 są równe O , dodawanie jest trywialne. Przypadek $P_1 = \pm P_2$ również może zostać rozpatrzony, ale pomijamy tutaj szczegóły.)

Możemy wyrazić P_1 i P_2 jako $(X_1/Z_1, Y_1/Z_1)$ i $(X_2/Z_2, Y_2/Z_2)$ we współrzędnych afinicznych, więc

$$\begin{aligned} P_3 &\stackrel{\text{def}}{=} P_1 + P_2 = m_2 - X_1/Z_1 - X_2/Z_2, m \cdot (X_1/Z_1 \\ &\quad - m_2 + X_1/Z_1 + X_2/Z_2) - Y_1/Z_1, 1 \end{aligned}$$

Gdzie

$$m = (Y_2/Z_2 - Y_1/Z_1)(X_2/Z_2 - X_1/Z_1) \quad 1 = (Y_2Z_1 - Y_1Z_2)(X_2Z_1 - X_1Z_2) \quad 1$$

i wszystkie obliczenia są wykonywane modulo p . Zauważ, że używamy współrzędnych rzutowych do przedstawienia P_3 , ustawiając $Z_3 = 1$ powyżej. Ale użycie współrzędnych rzutowych oznacza, że nie jesteśmy ograniczeni do $Z_3 = 1$. Mnożąc każdą współrzędną przez $Z_1Z_2(X_2Z_1 - X_1Z_2) = 0 \pmod{p}$, stwierdzamy, że P_3 można również przedstawić jako

$$P_3 = vw, u(v^2X_1Z_2 - w) \quad v^3Y_1Z_2, Z_1Z_2v^3 \quad (8.3)$$

Gdzie

$$\begin{aligned} u &= Y_2Z_1 - Y_1Z_2, v = X_2Z_1 - X_1Z_2, w = u^2Z_1Z_2 \\ &\quad - v^3 \quad 2v^2X_1Z_2. \end{aligned} \quad (8.4)$$

Warto zauważyć, że obliczenia w równaniach (8.3) i (8.4) można przeprowadzić bez konieczności wykonywania jakichkolwiek inwersji modułowych.

Właśnie dlatego, że punkty mają wiele reprezentacji we współrzędnych rzutowych, przy stosowaniu współrzędnych rzutowych mogą pojawić się pewne subtelności. (My

do tej pory wyraźnie zakładali, że elementy grupowe mają unikalną reprezentację jako ciągi bitów.) W szczególności punkt wyrażony we współrzędnych rzutowych może ujawnić pewne informacje o tym, jak ten punkt został uzyskany, co może zależeć od pewnych tajnych informacji. Aby temu zaradzić – a także ze względu na wydajność – do przesyłania i przechowywania punktów należy używać współrzędnych afinicznych, przy czym współrzędne rzutowe należy stosować jedynie jako pośrednią reprezentację w trakcie obliczeń (z punktami konwertowanymi na/ze współrzędnych rzutowych w początek/koniec obliczeń).

8.4 *Aplikacje kryptograficzne

Spędziliśmy sporo czasu omawiając teorię liczb i teorię grup oraz wprowadzając założenia dotyczące twardości obliczeniowej, które są powszechnie uważane za aktualne. Zastosowania tych założeń zajmiemy się przez resztę książki, ale tutaj podamy kilka krótkich przykładów.

8.4.1 Funkcje i permutacje jednokierunkowe

Funkcje jednokierunkowe są minimalnym prymitywem kryptograficznym i są zarówno niezbędne, jak i wystarczające do szyfrowania klucza prywatnego i kodów uwierzytelniających wiadomości. Pełniejsze omówienie roli funkcji jednokierunkowych w kryptografii znajduje się w rozdziale 7; tutaj podajemy jedynie definicję funkcji jednokierunkowych i pokazujemy, że ich istnienie wynika z liczbowych założeń dotyczących twardości, które widzieliśmy w tym rozdziale.

Nieformalnie funkcja f jest jednokierunkowa, jeśli można ją łatwo obliczyć, ale trudno ją odwrócić. Poniższy eksperyment i definicja, będące przekształceniem Definicji 7.1, formalizują to.

Doświadczenie odwracające InvertA,f (n):

1. Wybierz uniform $x \in \{0, 1\}^n$ i oblicz $y := f(x)$.
2. A ma na wejściu 1 n i y, a na wyjściu x.
3. Wynik eksperymentu wynosi 1 wtedy i tylko wtedy, gdy $f(x) = y$.

DEFINICJA 8.72 Funkcja $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ jest jednokierunkowa, jeśli spełnione są dwa warunki:

1. (Łatwe do obliczenia:) Istnieje algorytm czasu wielomianowego, który na wejściu x daje $f(x)$.
2. (Trudno odwrócić :) Dla wszystkich algorytmów ppt A istnieje znikoma funkcja zaniedbywanie w taki sposób, że $\Pr[\text{InvertA}, f(n) = 1] \geq \text{negl}(n)$.

Pokażemy teraz formalnie, że założenie o faktoringu implikuje istnienie funkcji jednokierunkowej. Niech Gen będzie algorytmem działającym w czasie wielomianowym, który na wejściu $1n$ daje wyniki (N, p, q) , gdzie $N = pq$ oraz p i q są n -bitowymi liczbami pierwszymi, z wyjątkiem tego, że prawdopodobieństwo jest znikome w n . (Używamy tutaj Gen zamiast GenModulus wyłącznie dla wygody zapisu). Ponieważ Gen działa w czasie wielomianowym, istnieje górną granicą wielomianu liczby losowych bitów używanych przez algorytm. Dla uproszczenia i zrozumienia głównych idei zakładamy, że Gen zawsze używa co najwyżej n losowych bitów na wejściu $1n$. W algorytmie 8.73 definiujemy funkcję $fGen$, która wykorzystuje swoje dane wejściowe jako losowe bity do uruchomienia Gen. Zatem $fGen$ jest funkcją deterministyczną, zgodnie z wymaganiami.

ALGORYTM 8.73 Obliczanie

algorytmów $fGen$

Dane wejściowe: Ciąg x o długości n

Dane wyjściowe: liczba całkowita N

$\text{oblicz } (N, p, q) := \text{Gen}(1n ; x) // \text{tj. uruchom}$

$\text{Gen}(1n)$ używając x jako losowego powrotu taśmy N

Jeśli problem faktoringu jest trudny w odniesieniu do Gen , wówczas intuicyjnie $fGen$ jest funkcją jednokierunkową. Z pewnością $fGen$ jest łatwe do obliczenia. Jeśli chodzi o trudność odwracania tej funkcji, należy zauważać, że następujące rozkłady są identyczne:

1. Moduł N wyprowadzany przez $fGen(x)$, gdy $x \in \{0, 1\}^n$ jest wybrane równomiernie.
2. Moduł N wyprowadzany przez (algorytm losowy) $\text{Gen}(1n)$.

Jeśli moduły N wygenerowane według drugiego rozkładu są trudne do rozłożenia na czynniki, to to samo dotyczy modułów N wygenerowanych według pierwszego rozkładu.

Co więcej, biorąc pod uwagę dowolny obraz wstępny x w odniesieniu do $fGen$ (tj. x , dla którego $fGen(x) = N$), zauważ, że nie wymagamy $x = x$, łatwo jest odzyskać współczynnik N , uruchamiając $\text{Gen}(1n; x)$ aby otrzymać (N, p, q) i wyprowadzić współczynniki p i q . Zatem znalezienie obrazu wstępnego N w odniesieniu do $fGen$ jest tak samo trudne, jak rozłożenie na czynniki N . Można to łatwo przekształcić w formalny dowód następującego twierdzenia:

TWIERDZENIE 8.74 Jeżeli problem rozkładu na czynniki jest trudny w odniesieniu do Gen , to $fGen$ jest funkcją jednokierunkową.

Permutacje jednokierunkowe

Możemy również użyć założeń teorii liczb, aby skonstruować rodzinę jednokierunkowych permutacji. Zaczynamy od ponownego przedstawienia definicji 7.2 i 7.3, specjalizujących się w przypadku permutacji:

DEFINICJA 8.75 Potrójny $\Pi = (\text{Gen}, \text{Samp}, f)$ probabilistycznych algorytmów wielomianowych w czasie jest rodziną permutacji, jeśli zachodzi następujący warunek:

1. Algorytm generowania parametrów Gen na wejściu 1^n wyprowadza parametry I za pomocą $|I| = r$. Każda wartość I definiuje zbiór DI , który stanowi dziedzinę i zakres permutacji (tj. bijekcji) $f_I : DI \rightarrow DI$.
2. Algorytm próbkowania Samp na wejściu I wyprowadza równomiernie rozłożony element DI .
3. Deterministyczny algorytm oceny f na wejściu $I \times DI$ wyprowadza elementy DI . Zapisujemy to jako $y := f_I(x)$.

Biorąc pod uwagę rodzinę funkcji Π , rozważ następujący eksperyment dla dowolnego algorytmu A i parametru n :

Doświadczenie odwracające InvertA, $\Pi(n)$:

1. Uruchomiono Gen(1^n) w celu uzyskania I , a następnie uruchomiono Samp(I) w celu wybrania jednolitego $x \in DI$. Na koniec obliczane jest $y := f_I(x)$.
2. A ma dane wejściowe I i y , a wyjściem jest x .
3. Wynik eksperymentu wynosi 1 wtedy i tylko wtedy, gdy $f_I(x) = y$.

DEFINICJA 8.76 Rodzina permutacji $\Pi = (\text{Gen}, \text{Samp}, f)$ jest jednokierunkowa, jeśli dla wszystkich probabilistycznych algorytmów wielomianowych A w czasie istnieje pomijalna funkcja negl taka, że

$$\Pr[\text{InvertA, } \Pi(n) = 1] = \text{negl}(n).$$

Biorąc pod uwagę GenRSA jak w Sekcji 8.2.4, Konstrukcja 8.77 definiuje rodzinę permutacji. Od razu widać, że jeśli problem RSA jest trudny w porównaniu z GenRSA, to ta rodzina jest jednokierunkowa. Można podobnie wykazać, że twardość problemu logarytmu dyskretnego w Z_p z pierwszym implikuje istnienie P , jednokierunkowej rodziny permutacji; patrz sekcja 7.1.2.

KONSTRUKCJA 8.77

Niech GenRSA będzie jak dawniej. Zdefiniuj rodzinę permutacji w następujący sposób:

- Gen: na wejściu 1^n , uruchom GenRSA(1^n), aby uzyskać (N, e, d) i wynik $J_a = N$, np. Ustaw $DI = Z \rightarrow N$.
- Próbka: na wejściu $I = N$, wybierz element jednolity $Z \rightarrow N$.
- f: na wejściu $I = N$, $e \in X \rightarrow Z \rightarrow N$, wyjście $[x \text{ mod } N]$.

Rodzina permutacji oparta na problemie RSA.

8.4.2 Konstruowanie odpornych na kolizje funkcji skrótu

Odporne na kolizje funkcje skrótu zostały wprowadzone w rozdziale 5.1. Chociaż konstrukcje odpornych na kolizje funkcji skrótu stosowanych w praktyce omówiliśmy w podrozdziale 6.3, nie widzieliśmy jeszcze konstrukcji, które można rygorystycznie opierać na prostszych założeniach. Pokazujemy tutaj konstrukcję opartą na założeniu logarytmu dyskretnego w grupach pierwszego rzędu. (Konstrukcja oparta na problemie RSA została opisana w ćwiczeniu 8.20.) Chociaż konstrukcje te są mniej wydajne niż funkcje mieszające stosowane w praktyce, są one ważne, ponieważ ilustrują możliwość osiągnięcia odporności na kolizję w oparciu o standardowe i dobrze zbadane założenia teorii liczb.

Niech G będzie algorytmem czasu wielomianowego, który na wejściu $1n$ wyprowadza (opis a) grupę cykliczną G , jej rząd q (przy $q = n$) i generator g . Tutaj również wymagamy, aby q było liczbą pierwszą, chyba że z znikomym prawdopodobieństwem. Funkcja mieszająca o stałej długości oparta na G jest podana w konstrukcji 8.78.

KONSTRUKCJA 8.78

Niech G będzie takie, jak opisano w tekście. Zdefiniuj funkcję skrótu o stałej długości (Gen, H) w następujący sposób:

- Gen: na wejściu $1n$, uruchom $G(1n)$, aby otrzymać (G, q, g) , a następnie wybierz a uniform $h \in G$. Wyjście $s := G, q, g, h$ jako klucz.
- H: mając klucz $s = G, q, g, h$ i wejście $(x_1, x_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$, wyjście $H(s, x_1, x_2) := g^{x_1} h^{x_2}$.

Funkcja mieszająca o stałej długości.

Należy zauważyć, że Gen i H można obliczyć w czasie wielomianowym. Zanim przejdziemy do dalszej analizy konstrukcji, poczynimy kilka uwag technicznych:

- Dla danego $s = G, q, g, h$, gdzie $n = q$, funkcję $H(s)$ opisuje się jako przyjmującą na wejściu elementy $\mathbb{Z}_q \times \mathbb{Z}_q$. Jednak $H(s)$ można postrzegać jako przyjmowanie ciągów bitów o długości $2 \cdot (n - 1)$ jako danych wejściowych, jeśli analizujemy dane wejściowe $x \in \{0, 1\}^{2(n-1)}$ jako dwa ciągi x_1, x_2 , każdy o długości $n - 1$, a następnie w naturalny sposób spojrzeć na x_1, x_2 jako elementy \mathbb{Z}_q .
- Wynik $H(s)$ jest podobnie określony jako element G , ale możemy go postrzegać jako ciąg bitów, jeśli ustalimy jakąś reprezentację G . Aby spełnić wymagania definicji 5.2 (która wymaga, aby długość wyjściowa była ustalona jako funkcją n) możemy w razie potrzeby uzupełnić dane wyjściowe.
- Biorąc pod uwagę powyższe, konstrukcja kompresuje dane wejściowe tylko dla grup G , w których elementy G mogą być reprezentowane przy użyciu mniej niż $2n - 2$ bitów. Odnosi się to zarówno do grup wyprowadzanych przez algorytm 8.65 (zakładając, że tak $N \in \mathbb{N}$, jest zwykle), jak i do grup krzywych eliptycznych, gdy stosowana jest kompresja punktowa. Uogólnienie konstrukcji 8.78

można użyć do uzyskania kompresji z dowolnego G , dla którego problem logarytmu dyskretnego jest trudny, niezależnie od liczby bitów wymaganych do reprezentowania elementów grupy; patrz ćwiczenie 8.21.

TWIERDZENIE 8.79 Jeśli problem logarytmu dyskretnego jest trudny w porównaniu z G , to Konstrukcja 8.78 jest odporną na kolizje funkcją mieszającą o stałej długości (podlega dyskusji dotyczącej kompresji powyżej).

DOWÓD Niech $\Pi = (\text{Gen}, \text{H})$ będzie takie jak w Konstrukcji 8.78 i niech A będzie probabilistycznym algorytmem wielomianowym w czasie

$$\epsilon(n) \stackrel{\text{def}}{=} \Pr[\text{Hash-collA}, \Pi(n) = 1]$$

(por. Definicja 5.2). Pokazujemy, jak A może zostać wykorzystane przez algorytm A do rozwiązania problemu logarytmu dyskretnego z prawdopodobieństwem sukcesu $\epsilon(n)$:

Algorytm A: Na wejściu algorytmu podano G, q, g, h .

1. Niech $s := G, q, g, h$. Uruchom $A(s)$ i uzyskaj dane wyjściowe x_1, x_2 .

2. Jeśli $x_1 = x_2$ i $H_s(x_1) = H_s(x_2)$ to: (a) Jeśli $h = 1$ zwróć 0. (b)

W przeciwnym razie ($h = 1$),

przeanalizuj x jako (x_1, x_2) i przeanalizuj x jako (x_1, x_2) , gdzie $x_1, x_2, x \in \mathbb{Z}_q$ i zwróć wynik $x_1 \cdot x_2$

$$(x_1 \cdot x_2) \equiv 1 \pmod{q} \quad (x_1 - 1) \cdot (x_2 - 1) \equiv 1 \pmod{q}.$$

Jest oczywiste, że A przebiega w czasie wielomianowym. Co więcej, dane wejściowe s przekazane A , gdy są uruchamiane jako podprogram przez A , są dystrybuowane dokładnie tak, jak w eksperymencie Hash-collA, Π dla tej samej wartości parametru bezpieczeństwa n . (Wejście do A jest generowane poprzez uruchomienie $G(1n)$ w celu uzyskania G, q, g , a następnie równomierne i losowe wybranie h

G. Dokładnie w ten sposób s jest generowane przez $\text{Gen}(1n)$. Zatem z prawdopodobieństwem dokładnie $\epsilon(n)$ doszło do kolizji; tj. $x_1 = x_2$ i $H_s(x_1) = H_s(x_2)$.

Twierdzimy, że za każdym razem, gdy dochodzi do kolizji, A zwraca poprawną odpowiedź $\log h$. Jeśli $h = 1$, to jest to oczywiście prawda (ponieważ w tym przypadku $\log h = 0$). Zakładając, że $h = 1$, oznacza to, że doszło do kolizji

$$\begin{aligned} H_s(x_1, x_2) &= H_s(x_1, x_2) \quad g^{x_1} h^{x_2} \\ 1 &\equiv g^{x_1} \cdot h^{x_2} \equiv g^{x_1} \cdot g^{x_2} \pmod{q} \end{aligned} \tag{8.5}$$

Zauważ, że $x \pmod{q}$ albo $\equiv 0 \pmod{q}$ (czyli $x_1 = x_2$) albo $\not\equiv 0 \pmod{q}$ (czyli $x_1 \neq x_2$). W drugim przypadku, istnieje odwrotność $\log h$. Podnieś do tego każdą stronę równania (8.5).

zdecydowanie $\equiv [(x_1 - 1)(x_2 - 1)]^{-1} \pmod{q}$

potęga daje:

$$(x_1 - 1)^{-1} \cdot x_2 \cdot x_2 \equiv g^{x_2} \pmod{q} \quad \text{zdecydowanie } 1 \equiv g^{x_2} \pmod{q}$$

więc wynik zwrócony przez A wynosi

$$\log h = [(x_1 - x_{-1}) \cdot \dots \mod q] = (x_1 - x_{-1}) \cdot (x_2 - x_3) \dots^1 \mod q.$$

Widzimy, że A poprawnie rozwiązuje problem logarytmu dyskretnego z prawdopodobieństwem dokładnie $\epsilon(n)$. Ponieważ z założenia problem logarytmu dyskretnego jest trudny w stosunku do G, dochodzimy do wniosku, że $\epsilon(n)$ jest nieistotne. ■

Stosując ćwiczenie 8.21 w połączeniu z transformacją Merkle'a – Damgåarda (patrz podrozdział 5.2) otrzymujemy:

TWIERDZENIE 8.80 Jeśli problem logarytmu dyskretnego jest trudny, wówczas istnieją funkcje mieszające odporne na kolizje.

Referencje i dodatkowe lektury

Książka Childsa [44] doskonale omawia teorię grup omawianą w tym rozdziale (i nie tylko), bardziej szczegółowo, ale na podobnym poziomie prezentacji.

Shoup [159] przedstawia bardziej zaawansowaną, ale wciąż przystępna analizę dużej części tego materiału, ze szczególnym uwzględnieniem aspektów algorytmicznych. (Nasze stwierdzenie postulatu Bertranda zostało zaczerpnięte z [159, Twierdzenie 5.8].) Stosunkowo łagodne wprowadzenia do algebry abstrakcyjnej i teorii grup, które wykraczają daleko poza to, na co mamy tu miejsce, są dostępne w książkach Fraleigha [67] i Hersteina [89]; zainteresowany czytelnik nie będzie miał trudności ze znalezieniem bardziej zaawansowanych tekstów z zakresu algebry, jeśli ma na to ochotę.

Pierwszy skuteczny test pierwszości przeprowadzili Solovay i Strassen [164]. Autorami testu Millera–Rabina są Miller [127] i Rabin [146]. Agrawal i in. odkryli deterministyczny test pierwszości. [5]. Obszerne badanie tego obszaru można znaleźć w Dietzfelbinger [57].

Problem RSA został publicznie przedstawiony przez Rivesta, Shamira i Adlemana [148], chociaż w 1997 roku ujawniono, że Ellis, Cocks i Williamson, trzej członkowie GCHQ (brytyjska agencja wywiadowcza), badali podobne pomysły – bez w pełni zdając sobie sprawę z ich znaczenia – w tajnym otoczeniu kilka lat wcześniej. Problemy logarytmu dyskretnego i problemów Diffiego–Hellmana po raz pierwszy rozważyli, przynajmniej pośrednio, Diffie i Hellman [58].

Większość metod leczenia krzywych eliptycznych wymaga od czytelnika zaawansowanej wiedzy matematycznej. Książka Silvermana i Tate'a [160] jest być może wyjątkiem. Jednakże, podobnie jak wiele książek na ten temat napisanych dla matematyków, ta książka zawiera niewielkie omówienie krzywych eliptycznych w ciałach skończonych, co jest najbardziej istotne w przypadku kriptografii. Tekst Washingtona-tona [176], choć nieco bardziej zaawansowany, zajmuje się głównie (ale nie wyłącznie)

z przypadkiem pola skończonego. Zagadnienia wdrożeniowe związane z kryptografią krzywych eliptycznych omówiono w Hankerson et al. [83]. Zalecane parametry dla krzywych eliptycznych oraz podgrup modulo a prime podaje NIST [132].

Konstrukcja odpornej na kolizje funkcji skrótu w oparciu o problem logarytmu dyskretnego wynika z [43], a wcześniejsza konstrukcja oparta na trudności rozkładu na czynniki jest podana w [81] (patrz także ćwiczenie 8.20).

Ćwiczenia

8.1 Niech G będzie grupą abelową. Udowodnić, że w G istnieje jednoznaczna tożsamość i że każdy element $g \in G$ ma swoją odwrotność.

8.2 Pokaż, że Twierdzenie 8.36 niekoniecznie zachodzi, gdy G jest nieskończone.

Wskazówka: rozważ zbiór $\{1\} \cup \{2, 4, 6, 8, \dots\} \subset \mathbb{R}$.

8.3 Niech G będzie grupą skońzoną, a $g \in G$. Pokaż, że g jest podgrupą G , $\langle g \rangle = \{g^0, g^1, \dots\}$. Czy zbiór $\{g^0\}$ koniecznie podgrupa G , gdy G jest nieskończone?

8.4 Pytanie to dotyczy funkcji Eulera ϕ .

- (a) Niech p będzie liczbą pierwszą, a $e > 1$ liczbą całkowitą. Pokaż, że $\phi(p^e) = p^{e-1} (p-1)$.
- (b) Niech p, q będą względnie pierwsze. Pokaż, że $\phi(pq) = \phi(p) \cdot \phi(q)$. (Możesz skorzystać z chińskiego twierdzenia o resztach.)
- (c) Udowodnić twierdzenie 8.19.

8.5 Oblicz dwie ostatnie cyfry (dziesiętne) liczby 31000 (ręcznie).

Wskazówka: odpowiedź brzmi $[31000 \bmod 100]$.

8.6 Oblicz $[1014 \ 800 \ 000 \ 002 \bmod 35]$ (ręcznie).

8.7 Oblicz $[4651 \bmod 55]$ (ręcznie) używając chińskiego twierdzenia o resztach.

8.8 Udowodnij, że jeśli G, H są grupami, to $G \times H$ są grupą.

8.9 Niech p, N będą liczbami całkowitymi, gdzie $p \mid N$. Udowodnić, że dla dowolnej liczby całkowitej X

$$[[X \bmod N] \bmod p] = [X \bmod p].$$

Pokaż, że dla kontrastu $[[X \bmod p] \bmod N]$ nie musi być równe $[X \bmod N]$.

8.10 Wniosek 8.21 pokazuje, że jeśli $N = pq$ i $ed = 1 \bmod \varphi(N)$, to $x \bmod N \equiv x \bmod p$. Pokaż, że wszystkie $x \in \mathbb{Z}_N^*$ mamy $(x^e)^d \equiv x \bmod N$ to obowiązuje dla wszystkich $x \in \{0, \dots, N-1\}$.

Wskazówka: Skorzystaj z chińskiego twierdzenia o resztach.

8.11 Uzupełnij szczegóły dowodu chińskiego twierdzenia o resztach, pokazując, że Z_N jest izomorficzny z $Z_s \times Z^*$.

8.12 W tym ćwiczeniu rozwijany jest efektywny algorytm sprawdzania, czy integer to potęga doskonała.

(a) Pokaż, że jeśli $N = N^e$ dla niektórych liczb całkowitych $N, e > 1$, to N . (b)

Mając N i e z $2 \leq N + 1$, pokaż, jak określić w czasie $\text{poli}(N)$ czy istnieje liczba całkowita N^e z $N^e = N$.

Wskazówka: użyj wyszukiwania binarnego.

(c) Mając N , pokaż, jak sprawdzić w czasie $\text{poli}(N)$, czy N jest potągą doskonałą.

8.13 Biorąc pod uwagę N i $a \in Z_N$, pokaż, jak sprawdzić w czasie wielomianowym, czy a jest mocnym dowodem na to, że N jest złożone.

8.14 Napraw N, e z $\text{gcd}(e, \phi(N)) = 1$ i załóżmy, że istnieje przeciwnik A działający w czasie t , dla którego

$$\Pr[A([x \text{ mod } N]) = x] = 0,01,$$

gdzie przyjmuje się prawdopodobieństwo jednolitego wyboru $x \in Z_N$, że N . Pokazywać możliwe jest skonstruowanie przeciwnika A, dla którego

$$\Pr[A([x \text{ mod } N]) = x] = 0,99$$

dla wszystkich x . Czas działania t A powinien być wielomianem w t i N .

Wskazówka: Wykorzystaj fakt, że $y^{1/e} \cdot r = (y \cdot r)^{1/e} \equiv 1 \pmod{N}$.

8.15 Formalnie zdefiniuj założenie CDH. Udowodnić, że twardość problemu CDH względem G implikuje twardość problemu logarytmu dyskretnego względem G i że twardość problemu DDH względem G implikuje twardość problemu CDH względem G .

8.16 Wyznacz punkty na krzywej eliptycznej $E : y^2 \equiv x^3 + 2x + 1 \pmod{Z_{11}}$.
znajduje się na tej krzywej?

8.17 Rozważmy grupę krzywych eliptycznych z Przykładu 8.67. (Patrz także Przykład 8.69.)
Oblicz $(1, 0) + (4, 3) + (4, 3)$ w tej grupie, najpierw przekształcając ją na współrzędne rzutowe, a następnie używając równań (8.3) i (8.4).

8.18 Udowodnij czwarte stwierdzenie Twierdzenia 8.68.

8.19 Czy poniższe zadanie można rozwiązać w czasie wielomianowym? Mając liczbę pierwszą p , wartość $x \in Z_p - \{1\}$, i $y := [g \cdot x \pmod{p}]$ (gdzie g jest jednolitą wartością $1/x \pmod{p}$) znajdź g , tj. oblicz y algorytmem p. Jeśli Twoja odpowiedź brzmi „tak”, podaj a), czasu wielomianowego. Jeśli Twoja odpowiedź brzmi „nie”, pokaż redukcję do jednego z założeń przedstawionych w tym rozdziale.

8.20 Niech GenRSA będzie takie jak w podrozdziale 8.2.4. Udowodnić, że jeśli problem RSA jest trudny w porównaniu z GenRSA , to Konstrukcja 8.81 jest odporną na kolizje funkcją skrótu o stałej długości.

KONSTRUKCJA 8.81

Zdefiniuj (Gen, H) w następujący

- sposób: • Gen: na wejściu, uruchom GenRSA($1n$), aby uzyskać N, e, d i wybierz $y \in Z_N$. Kluczem jest $s := N, e, y$.
- H: jeśli $s = N, e, y$, to H^s mapuje dane wejściowe $w \in \{0, 1\}^{3n}$ do wyjścia $w \in Z_N$. Niech $\hat{O}(x) = [x \text{ mod } N]$ i $f_1(x) = [y \cdot x]^{3n} \text{ mod } N$.
f Dla 3n-bitowego łańcucha $x = x_1 \cdots x_{3n}$, zdefiniuj

$$H^s(x) \stackrel{\text{def}}{=} f_{x_1}^{s_1} F_{x_2}^{s_2} \cdots 1 \cdots .$$

8.21 Rozważmy następujące uogólnienie konstrukcji 8.78:

KONSTRUKCJA 8.82

Zdefiniuj funkcję skrótu o stałej długości (Gen, H) w następujący sposób:

- (a) Gen: na wejściu $1n ht$, uruchom $G(1n)$, aby otrzymać $(G, q, h1)$, a następnie wybierz dany G . Wyjście $s := G, q, (h1, \dots, ht)$ jako klucz. $h2, \dots, (b) H$:

$$\text{klucz } s = G, q, (h1, \dots, ht) \text{ i wejście } (x_1, \dots, x_t) \stackrel{\text{def}}{=} z \underset{H}{\underset{x_i}{\underset{1 \text{ godz.}}{\underset{-}{\text{godz.}}}}}, \text{ wyjście } \underset{q}{\underset{x_i}{\underset{1 \text{ godz.}}{\underset{-}{\text{godz.}}}}}$$

(a) Udowodnić, że jeśli problem logarytmu dyskretnego jest trudny w stosunku do G i q jest liczbą pierwszą, to dla dowolnego $t = \text{poli}(n)$ konstrukcja ta jest odporna na kolizje funkcją mieszającą o stałej długości.

(b) Omów, w jaki sposób można zastosować tę konstrukcję do uzyskania kompresji niezależnie od liczby bitów potrzebnych do przedstawienia elementów G (o ile jest to wielomian w n).

Rozdział 9

*Algorytmy Faktoringu i Obliczanie logarytmów dyskretnych

W ostatnim rozdziale przedstawiliśmy kilka – większość – problemów z teorii liczb w szczególności rozłożenie na czynniki iloczynu dwóch dużych liczb pierwszych i obliczenie logarytmów dyskretnych w pewnych grupach - co jest powszechnie uważane za trudne. Jak tam zdefiniowany, oznacza to, że zakłada się, że nie ma algorytmów czasu wielomianowego dla tych problemów. To asymptotyczne pojęcie twardości jednak mówi niewiele o tym, jak ustawić parametr bezpieczeństwa —czasami nazywany kluczem długość, chociaż terminów nie można stosować zamiennie – aby osiągnąć jakiś pożądany, konkretny poziom bezpieczeństwa w praktyce. Właściwe zrozumienie tego problemu jest niezwykle ważne dla rzeczywistego wdrażania kryptosystemów opartych na te problemy. Ustawienie zbyt niskiego parametru bezpieczeństwa oznacza kryptosystem może być podatny na ataki skuteczniejsze niż oczekiwano; bycie zbyt konserwatywnym i ustawienie zbyt wysokiego parametru bezpieczeństwa zapewni dobre bezpieczeństwo, ale kosztem wydajności dla uczciwych użytkowników. Względna trudność różne problemy z teorii liczb mogą również odgrywać rolę w ustalaniu, które problemy, które można wykorzystać przede wszystkim jako podstawę do budowy kryptosystemów.

Podstawową kwestią jest oczywiście to, że wyszukiwanie metodą brute-force może nie być rozwiązaniem najlepszy algorytm rozwiązania danego problemu; zatem użycie długości klucza n nie, ogólnie rzec biorąc, zapewnia ochronę przed atakującymi działającymi przez $2n$ czasu. Kontrastuje to z ustawieniem klucza prywatnego, w którym najlepsze ataki na istniejące szyfry blokowe mają mniej więcej złożoność wyszukiwania metodą brute-force (ignorując obliczenia wstępne). W rezultacie długości kluczy używane w ustawieniu klucza publicznego są zwykle takie same znacznie większe niż te używane w ustawieniu klucza prywatnego.

Aby lepiej zrozumieć ten punkt, w tym rozdziale przyjrzymy się kilku algorytmom czasu niewielomianowego do rozkładu na czynniki i obliczania logarytmów dyskretnych, które są znacznie lepsze niż wyszukiwanie metodą brute-force. Celem jest jedynie dawanie przedsmak istniejących algorytmów rozwiązywania tych problemów, a także zapewnienie podstawowych wskazówek dotyczących ustawiania parametrów w praktyce. Stawiamy na wysoki poziom pomysłów i świadomie nie zajmujemy się wieloma ważnymi szczegółami na poziomie implementacji, którymi należałoby się zająć, gdyby te algorytmy miały działać być stosowane w praktyce. Koncentrujemy się również wyłącznie na algorytmach klasycznych, i odsyłamy czytelnika gdzie indziej w celu omówienia znanych algorytmów wielomianu kwantowego w czasie(!) do rozkładu na czynniki i obliczania logarytmów dyskretnych. (Ten była to kolejna świadoma decyzja, zarówno dlatego, że nie chcieliśmy zakładać wiedzy z zakresu mechaniki kwantowej ze strony czytelnika i dlatego komputery kwantowe wydają się mało prawdopodobne w najbliższej przyszłości.)

Czytelnik może również zauważyc, że opisujemy jedynie algorytmy do rozkładu na czynniki i obliczania logarytmów dyskretnych, a nie algorytmy do, powiedzmy, rozwiązywania problemów RSA lub decyzyjnych problemów Diffiego-Hellmana. Nasz wybór uzasadniają fakty, że najbardziej znane algorytmy rozwiązywania RSA wymagają rozkładu modułu na czynniki oraz (w grupach omawianych w podrozdziałach 8.3.3 i 8.3.4) najbardziej znane podejścia do rozwiązywania decyzyjnego problemu Diffiego-Hellmana obejmują obliczenia logarytmów dyskretne.

9.1 Algorytmy faktoringu

W całym tekście zakładamy, że $N = pq$ jest iloczynem dwóch różnych liczb pierwszych, gdzie $p < q$. Najbardziej będzie nas interesował przypadek, gdy p i q mają tę samą (znaną) długość n , a zatem $n = \Theta(\log N)$.

Będziemy często używać chińskiego twierdzenia o resztach wraz z notacją rozwiniętą w rozdziale 8.1.5. Chińskie twierdzenie o resztach to stwierdza

$$ZN Zp \times Zq \models Z \quad N \equiv_p^x Zq,$$

z izomorfizmem określonym przez $f(x) = ([x \bmod p], [x \bmod q])$. Fakt, że f jest izomorfizmem oznacza w szczególności, że daje bijekcję pomiędzy elementami $x \in ZN$ i parami $(xp, xq) \in Zp \times Zq$. Zapisujemy $x \equiv (xp, xq)$, gdzie $xp = [x \bmod p]$ i $xq = [x \bmod q]$, aby oznaczyć tę bijekcję.

Przypomnijmy sobie z sekcji 8.2, że dzielenie próbne —trywialna metoda faktoryzacji metodą brutalnej siły —znajduje współczynnik o danej liczbie N w czasie $O(N^{1/2} \cdot \text{polilog}(N))$. (Jest to algorytm działający w czasie wykładowiczym, ponieważ rozmiar danych wejściowych jest długością binarnej reprezentacji N , tj. $N = O(\log N)$).¹ Omówimy trzy algorytmy faktoringu o lepszej wydajności:

- Metoda p-1 Pollarda jest skuteczna, jeśli $p-1$ ma tylko „małe” czynniki pierwsze.
- Metoda rho Pollarda ma zastosowanie do dowolnego N . (W związku z tym nazywa się ją algorytmem faktoryzacji ogólnego przeznaczenia). Czas jej wykonania dla N postaci omówionej na początku tej sekcji wynosi $O(N^{1/4} \cdot \text{polilog}(N))$.
Należy zauważyć, że jest to nadal wykładowicze w n , długości N .
- Algorytm sita kwadratowego jest algorytmem faktoryzacji ogólnego przeznaczenia, który działa w czasie podwykładowiczym na długości N . Dajemy ogólny przegląd działania tego algorytmu, ale szczegółowo są nieco złożone i wykraczają poza zakres ta książki.

¹ Zatem czas działania $O(N^{1/2}) = O(N)$ jest wykładowczy, czas działania $O(\log N)$ jest wielomianem.

Najszyszybszym znanym algorytmem faktoringu ogólnego przeznaczenia jest sító pola liczb ogólnych. Z heurystycznego punktu widzenia algorytm ten uwzględnia swoje dane wejściowe N w oczekiwany czasie $2O((\log N)^{1/3} 2^{2/3} \cdot (\log \log N))$, który jest podwykładniczy na długości N .

9.1.1 Algorytm p - 1 Pollarda

Jeśli $N = pq$ i $p-1$ ma tylko „małe” czynniki pierwsze, algorytm p-1 Pollarda może zostać użyty do efektywnego rozkładu na czynniki N . Podstawowa idea jest prosta. Niech B będzie liczbą całkowitą, dla której $(p-1) | B$ i $(q-1) | B$; poniżej opisujemy szczegółowo, jak znaleziono takie B . Powiedzmy $B = y \cdot (p-1)$ dla pewnej liczby całkowitej y . Wybierz jednorodne $x \in Z_N$ i oblicz $y := [x^{-B} \mod p, x^{-B} \mod q]$. (Zauważ, że y może być obliczone przy użyciu algorytmu efektywnego potęgowania Załącznika B.2.3.)

Ponieważ $1 \equiv (1, 1)$ mamy

$$\begin{aligned} y &= [x^{-B} \mod p, x^{-B} \mod q] \\ &= (x^{-B} \mod p, x^{-B} \mod q) \\ &= ((x^{-p-1} \mod p, x^{-1} \mod p), x^{-B} \mod q) \\ &= (0, [x^{-B} \mod q]) \end{aligned}$$

korzystając z Twierdzenia 8.14 i faktu, że rzad Z_N , który z $\mod p$ wynosi $p-1$. Pokazujemy poniżej dużym prawdopodobieństwem $x^{-B} \mod q \equiv 1 \mod q$. Zakładając, że tak jest, otrzymaliśmy liczbę całkowitą y , dla której

$$y \equiv 0 \mod p, \text{ ale } y \equiv 0 \mod q;$$

czyli $p | y$, ale $q | y$. To z kolei implikuje, że $\gcd(y, N) = p$. Zatem proste obliczenie \gcd (które można wykonać wydajnie, jak opisano w dodatku B.1.2) daje czynnik pierwszy N .

ALGORYTM 9.1 Algorytm Pollarda p-1 dla faktoringu

Wejście: liczba całkowita N

Wynik: Nietrywialny współczynnik N

```

x ∈ Z_N
y := [x^{-B} mod N]
// B jak w tekście p :=
gcd(y, N) if p
{1, N} return p
  
```

Załóżmy najpierw, że algorytm działa (z dużym prawdopodobieństwem). Przypuszczać $(p-1) | B$ ale $(q-1) | B$. W takim przypadku, o ile $xq \stackrel{\text{def}}{=} [x \mod q]$ wynosi $a = 1 \mod q$. generator Z_N mamy $xq^{-1} \mod q$ (Wynika to z Twierdzenia 8.52.)

Pozostaje zbadać prawdopodobieństwo, że xq jest generatorem. Tutaj polegamy

niektóre wyniki przedstawiono w Załączniku B.3.1. Ponieważ q jest liczbą pierwszą, Z jest grupą cykliczną rzędu $q - 1$, która ma dokładnie $\varphi(q - 1)$ generatory (por. Twierdzenie B.16). Jeśli x zostanie wybrane jednolicie z Z . (Jest to konsekwencja faktu, że istnieje równoznaczne rozłożenie reziduowe względem q .) Zatem prawdopodobieństwo, że xq jest generatorem, mieści się w przedziale $Z \cap N$ o $\varphi(q - 1) / q - 1$

$$\frac{1}{q-1}$$

$= \Omega(1/\log q) = \Omega(1/n)$ (por. Twierdzenie B.15). Wiele wartości x może wybrać, aby zwiększyć prawdopodobieństwo sukcesu.

Pozostaje nam problem znalezienia B takiego, że $(p - 1) | B$ ale $(q - 1) \nmid B$. $k = n/\log p$ Jedną z możliwości $=$ dla pewnego k , gdzie p_i oznacza $i=1$ do i -prim ($p_1 = 2, p_2 = 3, p_3 = 5, \dots$), a n oznacza długość str. $n/\log p$ to największa potęga p_i , która może podzielić $p - 1$. Jeśli i

że str

1 można zapisać jako $\prod_{i=1}^k p_i^{e_i}$ gdzie $e_i > 0$ (to znaczy, jeśli największą liczbą pierwszą p wspólny $p - 1$ mniejszy od p_k , będziemy mieli $(p - 1) | B$. Przeciwnie, jeśli $q - 1$ ma jakikolwiek czynnik pierwszy większy niż p_k , to $(q - 1) \nmid B$.

Wybranie większej wartości k zwiększa B , a tym samym wydłuża czas działania algorytmu (który wykonuje modułowe potęgowanie do potęgi B).

Większa wartość k zwiększa również prawdopodobieństwo, że $(p - 1) | B$, ale jednocześnie zmniejsza prawdopodobieństwo, że $(q - 1) \nmid B$. Możliwe jest oczywiście wielokrotne uruchomienie algorytmu przy wielokrotnych wyborach dla k .

Algorytm $p - 1$ Pollarda zostaje udaremiony, jeśli zarówno $p - 1$, jak i $q - 1$ mają jakiekolwiek duże czynniki pierwsze. (Dokładniej, algorytm nadal działa, ale tylko dla B tak dużego, że algorytm staje się niepraktyczny.) Jeśli p i q są jednolitymi n -bitowymi liczbami pierwszymi, to jest mało prawdopodobne, aby $p - 1$ lub $q - 1$ miały tylko małą liczbę pierwszą czynniki. Niemniej jednak, podczas generowania modułu $N = pq$ do zastosowań kryptograficznych, p i q są czasami wybierane jako mocne liczby pierwsze. (Przypomnijmy, że p jest silną liczbą pierwszą, jeśli $(p - 1)/2$ jest również liczbą pierwszą.) Wybieranie p i q w ten sposób jest znacznie mniej efektywne niż zwykłe wybieranie p i q jako dowolnych (losowych) liczb pierwszych. Ponieważ i tak dostępne są lepsze algorytmy faktoringu (jak zobaczymy poniżej) oraz w związku z powyższą obserwacją, obecny konsensus jest taki, że dodatkowy koszt obliczeniowy generowania p i q jako silnych liczb pierwszych nie daje żadnych zauważalnych korzyści w zakresie bezpieczeństwa.

9.1.2 Algorytm Rho Pollarda

Algorytm rho Pollarda można wykorzystać do rozłożenia na czynniki dowolnej liczby całkowitej $N = pq$; w tym sensie jest to algorytm faktoringu ogólnego przeznaczenia. Heurystycznie algorytm uwzględnia N ze stałym prawdopodobieństwem w czasie $O(N^{1/4} \cdot \text{polilog}(N))$; jest to nadal wykładnicze, ale stanowi ogromną poprawę w porównaniu z podziałem próbny.

Podstawową ideą tego podejścia jest znalezienie różnych wartości $x, x' \in Z$ które są N równe modulo p (tj. dla których $x \equiv x' \pmod{p}$); nazwać taką parę dobrą. Zauważ, że dla dobrych par x, x' utrzymuje się, że $\gcd(x - x', N) = p$ (ponieważ $x \equiv x' \pmod{N}$), więc obliczenie \gcd daje nietywialny wspólny czynnik.

Jak znaleźć dobrą parę? Powiedzmy, że wybieramy wartości $x(1), \dots, x(k)$ uni-gdzie $k = 2n/2 = O(n)$ formalnie od $Z \equiv N \pmod{p}$. Przeglądanie ich w języku chińskim-

ALGORYTM 9.2 Algorytm

rho Pollarda dla faktoringu

Dane wejściowe: Liczba całkowita N , iloczyn dwóch n-bitowych liczb pierwszych

Wynik: Nietrywialny współczynnik N

$x^{(0)} \in \mathbb{Z}_N, \quad x := x^{(0)}$

dla $i = 1$ do $2n/2$:

$x := F(x) \quad x :=$

$F(F(x)) \quad p := \gcd(x$

$\quad \quad \quad , N)$

jeśli $p \in \{1, N\}$ zwróć p i zatrzymaj się

pozostała reprezentacja jako $(x^{(1)}, \dots, x^{(k)})$, mamy, że każde $x \in \mathbb{Z}_p$,
 $x^{(i)} \stackrel{\text{def}}{=} [x(i) \bmod p]$ jest jednolite w \mathbb{Z}_p . (Wynika to z bijektywności pomiędzy.) Zatem
 $\exists p \in \mathbb{Z}_q$ korzystając z granicy urodzin z Lematu A.16, widzimy, że
 (i) z dużym prawdopodobieństwem istnieją odrębne $i, j \in \mathbb{Z}_p$ takie, że $x^{(i)} = x^{(j)}$ lub równoważnie $(i) \equiv (j) \pmod{p}$, z
 $x^{(i)} = x^{(j)} \bmod p$. Ponadto lemat A.15 pokazuje, że $x^{(i)} = x^{(j)}$ wyjątkiem znikomego
prawdopodobieństwa. Zatem z dużym prawdopodobieństwem otrzymujemy dobrą parę $x^{(i)}$, którą można
 $x^{(i)}$, wykorzystać do znalezienia nietrywialnego współczynnika N , jak omówiono wcześniej. w czasie $O(\sqrt{p})$
Możemy wygenerować $k = O(\sqrt{p})$ jednorodnych elementów $Z_N = \{x \in \mathbb{Z} : x^2 \equiv 1 \pmod{N}\}$. Jednakże
testowanie wszystkich par elementów w celu zidentyfikowania dobrej pary $= O(p) = O(N^{1/2})$ czasu! (Zauważ,
dobrą parę. Zamiast $O(N^{1/2})$ że ponieważ p wymagałoby jawności, a następnie posortuj, aby znaleźć
nieznane, nie możemy po prostu obliczyć $x^{(1)}, \dots, x^{(k)}$ obliczyć $\gcd(x^{(i)}, x^{(j)})$
 $x^{(i)} \in \mathbb{Z}_N$, aby sprawdzić, czy daje to nietrywialny współczynnik N .) Bez dalszych
optymizacji nie będzie to lepsze niż dzielenie próbne.

Pomysł Pollarda polegał na zastosowaniu techniki, którą widzieliśmy w sekcji 5.4.2 w
kontekście ataków urodzinowych na małą przestrzeń. W szczególności obliczamy sekwencję
 $x^{(1)}, x^{(2)}, \dots$ pozwalając, aby każda wartość była funkcją poprzedniej; tj. my $x^{(0)} \in \mathbb{Z}_N$
napraw jakąś funkcję $F: \mathbb{Z}_N \rightarrow \mathbb{Z}_N$, wybieramy jednorodny $x^{(0)}$ i następnie ustalamy
 $x^{(i)} := F(x^{(i-1)})$ dla $i = 1, \dots, k$. Wymagamy, aby F miał tę własność, że jeśli $x = x \bmod p$,
 $F(x) = F(x) \bmod p$; gwarantuje to, że gdy wystąpi równoważność modulo p , będzie
się ona utrzymywać. (Standardowym wyborem jest $F(x) = [x^2 + 1 \bmod N]$, ale każdy wielomian
 F będzie miał tę właściwość.) Jeśli modelujemy F jako funkcję losową (która działa
heurystycznie), to z dużym prawdopodobieństwem istnieje dobra para w pierwszych k
elementach tego ciągu. Postępując mniej więcej jak w algorytmie 5.9 z rozdziału 5.4.2,
możemy wykryć dobrą parę (jeśli taka istnieje) używając jedynie obliczeń $O(k) \gcd$; patrz
Algorytm 9.2. Oprócz skrócenia czasu działania pomysł Pollarda drastycznie zmniejsza
również ilość potrzebnej pamięci.

9.1.3 Algorytm sita kwadratowego

Algorytm rho Pollarda jest lepszy niż dzielenie próbne, ale nadal działa w czasie
wykładniczym. Algorytm sita kwadratowego działa w czasie subwykładniczym. To

był najszybszym znanym algorytmem faktoringu aż do początku lat 90. XX wieku i pozostało wybrany algorytm faktoryzacji dla liczb o długości do około 300 bitów. My opisz ogólne zasady algorytmu, ale ostrzegaj o tym czytelniku pominięto kilka ważnych szczegółów.

Element z \mathbb{Z}_N jest resztą kwadratową modulo N, jeśli istnieje $x \in \mathbb{Z}$ takie, że $x^2 \equiv z \pmod{N}$; w tym przypadku mówimy, że x jest pierwiastkiem kwadratowym z z. The za punkt wyjścia przyjmujemy następujące obserwacje:

- Jeśli N jest iloczynem dwóch różnych, nieparzystych liczb pierwszych, to każda liczba kwadratowa reszta modulo N ma dokładnie cztery pierwiastki kwadratowe. (Patrz rozdział 13.4.2.)
- Biorąc pod uwagę $x, y \in \mathbb{Z}^2$ takie, że $x^2 \equiv y^2 \pmod{N}$, jest to możliwe obliczyć nietrywialny współczynnik N w czasie wielomianowym. Dzieje się tak z cnoty faktu, że $x^2 - y^2 \equiv 0 \pmod{N}$ oznacza

$$x^2 - y^2 \equiv 0 \pmod{N},$$

i tak $N \mid (x - y)(x + y)$. Jednakże $N \nmid (x - y)$ i $N \nmid (x + y)$, ponieważ $x \equiv \pm y \pmod{N}$. Zatem musi być tak, że $\gcd(x - y, N)$ jest równe jednemu z czynników pierwszych N. (Zobacz także Lemat 13.35.)

Algorytm sita kwadratowego próbuje wygenerować x, y za pomocą $x^2 \equiv y^2 \pmod{N}$ i $x \equiv \pm y \pmod{N}$. Naiwny sposób na zrobienie tego - który stanowi podstawę starszego algorytmu faktoringu ze względu na Fermata - polega na wybraniu $x \in \mathbb{Z}_N$, obliczając $q := [x^2 \pmod{N}]$, a następnie sprawdź, czy q jest kwadratem liczb całkowitych (tj. bez modułu redukcji N). Jeśli tak, to $q = y^2$ dla pewnej liczby całkowitej y i tak $2x - 2y \equiv 0 \pmod{N}$. Niestety prawdopodobieństwo, że $[x^2 \pmod{N}]$ jest kwadratem jest tak niska, że proces ten należy powtarzać wykładniczo wiele razy.

Znaczącą poprawę uzyskuje się poprzez wygenerowanie ciągu wartości $q_1 := [x^2 \pmod{N}]$, ... oraz identyfikowanie podzbioru tych wartości, których iloczyn jest kwadratem po liczbach całkowitych. W algorytmie sita kwadratowego tak jest można osiągnąć w dwóch następujących krokach:

Krok 1. Napraw pewne ograniczenie B. Założmy, że liczba całkowita jest gładka B, jeśli jest w całości liczbą pierwszą i jej czynniki są mniejsze lub równe B. W pierwszej fazie algorytmu mamy szukaj liczb całkowitych w postaci $q_i = [x^2 \pmod{N}]$, które są gładkie i współczynnik B ich. (Chociaż rozkład na czynniki jest trudny, znajdowanie i rozkładanie na czynniki liczb B-gładkich jest wykonalne, gdy B jest wystarczająco małe.) Te $\{x_i\}$ są wybierane poprzez próbę $x = \overline{N+1}, \overline{N+2}, \dots$; zapewnia to nietrywialną redukcję modulo N (ponieważ $x > \sqrt{N}$ i ma tę zaletę, że $q_i = [x^2 \pmod{N}] = x^2 \pmod{N}$ jest „małe” więc q z większym prawdopodobieństwem będzie gładkie B).

Niech $\{p_1, \dots, p_k\}$ będzie zbiorem liczb pierwszych mniejszych lub równych B. Raz znaleźliśmy i rozłożyliśmy na czynniki B-gładkie $\{q_i\}$, jak opisano powyżej, mamy a

układ równań postaci:

$$\begin{aligned}
 q_1 &= [x^2 \mod N] = \prod_{j=1}^k p_i^{e_{1,i}} \\
 &\vdots \\
 q &= [x^2 \mod N] = \prod_{j=1}^k p_i^{e_{i,i}}.
 \end{aligned} \tag{9.1}$$

(Zauważ, że powyższe równania dotyczą liczb całkowitych.)

Krok 2. Następnie chcemy znaleźć jakiś podzbiór $\{q_j\}$, którego iloczynem jest kwadrat. Jeśli pomnożymy jakiś podzbiór S z $\{q_j\}$, zobaczymy, że wynik

$$Z = \prod_{j \in S} q_j = \prod_{j=1}^k \prod_{i \in \text{Liczby}_{q_j}} j^{\sum_{t=1}^k S_{ej,tj}}$$

jest kwadratem wtedy i tylko wtedy, gdy wykładnik każdej liczby pierwszej p_i jest parzysty. Sugeruje to, że o wykładniki $\{e_{j,i}\}$ w równaniu (9.1) dbamy tylko modulo 2; co więcej, możemy użyć algebry liniowej, aby znaleźć podzbiór $\{q_j\}$, którego „wektory wykładnicze” sumują się do wektora zerowego modulo 2.

Bardziej szczegółowo: jeśli zmniejszymy wykładniki w równaniu (9.1) modulo 2, to uzyskać macierz 0/1 Γ podaną przez

$$\begin{array}{ll}
 \gamma_1, \gamma_1, 2 \cdots \gamma_1, k & [e_{1,1} \mod 2] [e_{1,2} \mod 2] \cdots [e_{1,k} \mod 2] \\
 \vdots \quad \vdots \quad \ddots \quad \vdots \quad \stackrel{\text{def}}{=} & \vdots \quad \vdots \quad \ddots \quad \vdots \\
 \gamma_1, \gamma_2, \dots, \gamma_k & [e_{1,1} \mod 2] [e_{1,2} \mod 2] \cdots [e_{1,k} \mod 2]
 \end{array}$$

Jeśli $k = k + 1$, to Γ ma więcej wierszy niż kolumn i musi istnieć jakiś niepusty podzbiór S wierszy, których suma daje 0-wektor modulo 2. Taki podzbiór można efektywnie znaleźć za pomocą algebry liniowej. Następnie:

$$Z \stackrel{\text{def}}{=} \prod_{j \in S} q_j = \prod_{j=1}^k \prod_{i \in \text{Liczby}_{q_j}} j^{\sum_{t=1}^k S_{ej,tj}} = \prod_{j=1}^k \left(\prod_{i \in \text{Liczby}_{q_j}} e_{j,i} \right)^2,$$

korzystając z faktu, że wszystkie $\sum_{i \in \text{Liczby}_{q_j}} e_{j,i}$ są parzyste. Od

$$Z = \prod_{j \in S} q_j = \prod_{j \in S} \prod_{i \in \text{Liczby}_{q_j}} e_{j,i}^2 = \prod_{j \in S} x_j^2 \mod N,$$

otrzymaliśmy dwa pierwiastki kwadratowe (modulo N) z x . Chociaż nie ma gwarancji, że te pierwiastki kwadratowe umożliwiają faktoryzację N (z powodów

omówione na początku tej sekcji), heurystycznie robią to ze stałym prawdopodobieństwem. Biorąc $> k + 1$, możemy otrzymać wiele podzbiorów S o pożądanej właściwości i spróbować rozłożyć N na czynniki, korzystając z każdej możliwości.

Przykład 9.3

Weźmy $N = 377753$. Mamy $6647 = [6202 \bmod N]$ i możemy rozłożyć na czynniki 6647 (przez liczby całkowite, bez żadnej redukcji modułowej) jako

$$6202 \bmod N = 6647 = 172 \cdot 23.$$

Podobnie,

$$6212 \bmod N = 24 \cdot 17 \cdot 29 \quad 6452$$

$$\bmod N = 27 \cdot 13 \cdot 23 \quad 6552 \bmod$$

$$N = 23 \cdot 13 \cdot 17 \cdot 29.$$

Widzimy to, jeśli nasz podzbiór S obejmuje wszystkie cztery powyższe równania

$$\begin{aligned} 6202 \cdot 6212 \cdot 6452 \cdot 6552 &= 214 \cdot 132 \cdot 174 \cdot 232 \cdot 292 \bmod N \\ [620 \cdot 621 \cdot 645 \cdot 655 \bmod N] &\quad 2^7 = 2 \cdot 13 \cdot 172 \cdot 23 \cdot 29 \bmod N \quad 2 \bmod N \\ 1271942 &= 453352 \bmod N, \end{aligned}$$

gdzie $127194 = \pm 45335 \bmod N$. Obliczenie $\gcd(127194 - 45335, 377753) = 751$ daje nietrywialny współczynnik N .

Czas pracy. Wybór większej wartości B zwiększa prawdopodobieństwo, że jednolita wartość $q = [x \bmod N]$ będzie B -gładka; z drugiej strony oznacza to, że musimy częściej pracować, aby zidentyfikować i rozłożyć na czynniki liczby B -gładkie i będziemy musieli znaleźć ich więcej (ponieważ potrzebujemy $> k$, gdzie k jest liczbą liczb pierwszych mniejszą lub równą B). Oznacza to również, że macierz Γ będzie większa, a więc krok liniowo-algebraiczny będzie wolniejszy. Wybór optymalnej wartości B daje algorytm, który (przynajmniej heurystycznie) uwzględnia N w czasie $O(\sqrt{\log N \log \log N})$

. (W rzeczywistości stały składnik wykładnika można wyznaczyć dość precyzyjnie.) Dla naszych celów ważnym punktem jest to, że jest to współczynnik podwykładniczy na długości N .

9.2 Algorytmy obliczania logarytmów dyskretnych

Niech G będzie grupą znanego rzędu q . Przykład problemu logarytmu dyskretnego w G określa podstawę $g \in G$ (która nie musi być generatorem G) i element $h \in g$, podgrupę generowaną przez h ; celem jest znalezienie x

takie, że $g = \tilde{1}$. (Zobacz rozdział 8.3.2.) Rozwiązywanie x nazywa się logarymem dyskretnym h względem g . Trywialne wyszukiwanie x metodą brute-force można przeprowadzić w czasie $|g| - q$ (po prostu wypróbowując wszystkie możliwe wartości), dlatego interesują nas tylko algorytmy, których czas działania jest lepszy.

Algorytmy rozwiązywania problemu logarytmu dyskretnego dzielą się na dwie kategorie: te, które są ogólne i mają zastosowanie do dowolnych grup, oraz te, które są dostosowane do pracy dla określonej klasy grup. Zaczynamy od omówienia trzech ogólnych algorytmów:

- Gdy rząd grupowy q nie jest liczbą pierwszą i rozkład na czynniki q jest znany lub łatwy do ustalenia, algorytm Pohliga–Hellmana redukuje problem znajdowania logarytmów dyskretnych w G do problemu znajdowania logarytmów dyskretnych w podgrupach rzędu pierwszego G . Z grubsza rzecz biorąc, efekt jest taki, że złożoność rozwiązania logarytmu dyskretnego w grupie rzędu q nie jest większa niż złożoność rozwiązania logarytmu dyskretnego w grupie rzędu q , gdzie q jest największym podziałem pierwszym q . Wyjaśnia to preferencję stosowania grup pierwszego rzędu (por. sekcja 8.3.2).
- Metoda małego kroku/gigantycznego kroku, według Shanksa, oblicza logarytm dyskretny w grupie rzędu q w czasie $O(\sqrt{q} \cdot \text{polilog}(q))$ i przechowuje elementy grupy $O(\sqrt{q})$.
- Algorytm rho Pollarda umożliwia także obliczanie logarytmów dyskretnych w czasie $O(\sqrt{q} \cdot \text{polylog}(q))$, ale przy wykorzystaniu stałej pamięci. Można to postrzegać jako wykorzystanie powiązania między problemem logarytmu dyskretnego a hashowaniem odpornym na kolizje, co widzieliśmy w podrozdziale 8.4.2. Opiszemy inny algorytm, który jaśniejsze ilustruje to połączenie.

Mogą wykazać, że złożoność czasowa dwóch ostatnich algorytmów jest optymalna w przypadku algorytmów generycznych. Zatem, aby mieć nadzieję na lepsze działanie, musimy przyjrzeć się algorytmom dla konkretnych grup, które wykorzystują reprezentację elementów grup w tych grupach, gdzie przez „reprezentację” rozumiemy sposób, w jaki elementy grupy są kodowane jako ciągi bitów.

Ten punkt wymaga dyskusji. Z matematycznego punktu widzenia dowolne dwie grupy cykliczne tego samego rzędu są izomorficzne, co oznacza, że grupy są identyczne aż do „zmiany nazw” elementów grupy. Jednak z obliczeniowego/algorytmicznego punktu widzenia ta „zmiana nazwy” może mieć znaczący wpływ. Rozważmy na przykład grupę cykliczną Z_q liczb całkowitych $\{0, \dots, q-1\}$ w ramach dodawania modulo q . Obliczanie logarytmów dyskretnych w tej grupie jest trywialne. Powiedzmy, że mamy $g, h \in Z_q$ z generatorem g (więc $g = 0$) i chcemy znaleźć x takie, że $x \cdot g = h \pmod{q}$. Ponieważ $g = 0$, mamy $\gcd(g, q) = 1$, więc g ma multiplikatywną odwrotność g^{-1} modulo q . Co więcej, można je efektywnie obliczyć, jak opisano w Załączniku B.2.2. Ale wtedy $x = h \cdot g^{-1} \pmod{q}$ jest pożądanym rozwiązaniem. Należy zauważyć, że formalnie x oznacza tutaj liczbę całkowitą, a nie element grupy — w końcu operacja na grupie polega na dodawaniu, a nie mnożeniu. Niemniej jednak przy rozwiązywaniu logarytmu dyskretnego

problemu w Z_q możemy wykorzystać fakt, że na elementach tej grupy można zdefiniować inną operację (mianowicie mnożenie).

Zwracając się do grup o znaczeniu kryptograficznym, skupiamy naszą uwagę na (podgrupach) Z dla $p \neq \text{prime}$. (Zobacz rozdział 8.3.3.) Podajemy ogólny przegląd algorytmu rachunku indeksowego służącego do rozwiązywania problemu logarytmu dyskretnego w takich grupach w czasie subwykładniczym. Obecnie najbardziej znanym algorytmem dla tej klasy grup jest sító pól liczbowych, które² znane jest również obliczanie $1/3 \cdot 2/3 \cdot (\log \log p)$. Algorytmy subwykładnicze dla logarytmów dyskretnych w multiplikatywnych podgrupach ciał skończonych o dużej charakterystyce. Na początku 2013 r. poczyniono znaczne postępy w algorytmach obliczania logarytmów dyskretnych w multiplikatywnych podgrupach ciał skończonych o małych charakterystycznych; rozsądne wydaje się unikanie stosowania takich grup do zastosowań kryptograficznych.

Co ważne, nie są znane żadne algorytmy subwykładnicze do obliczania logarytmów dyskretnych w pewnych grupach krzywych eliptycznych. Oznacza to, że dla danego poziomu bezpieczeństwa możemy używać grup mniejszego rzędu, pracując w grupach krzywych eliptycznych w porównaniu, powiedzmy, pracując w Z_p , co skutkuje schematami kryptograficznymi o lepszej wydajności asymptotycznej.

9.2.1 Algorytm Pohliga–Hellmana

Algorytm Pohliga – Hellmana można wykorzystać do przyspieszenia obliczeń logarytmów dyskretnych w grupie G , gdy znane są nietrywialne czynniki rzędu grupowego q . Przypomnijmy, że rzząd elementu g , który oznaczamy tutaj przez $\text{ord}(g)$, jest najmniejszą dodatnią liczbą całkowitą i dla której $g^{\text{ord}(g)} = 1$. Będziemy potrzebować następującego lematu:

LEMMA 9.4 Niech $\text{ord}(g) = q$ i powiedzmy $p \mid Q$. Wtedy $\text{ord}(g|p) = q/p$.

DOWÓD Ponieważ $(g|p)^{q/p} = 1$ f_{zg}d g|p jest co najwyżej q/p . Niech $i > 0$ pi będzie takie, że $(g|p)^i = 1$. Wtedy $g = 1$ i ponieważ q jest rzędem g , musimy mieć $pi \mid q$ lub równoważnie $i \mid q/p$. Rząd $g|p$ jest więc dokładnie q/p . ■

Skorzystamy także z uogólnienia chińskiego twierdzenia o resztach: jeśli $q = \prod_{i=1}^k q_i$ i $\text{gcd}(q_i, q_j) = 1$ dla wszystkich $i \neq j$ wtedy

$$\sum_{i=1}^k Z_{q_i} \times \cdots \times Z_{q_k} \equiv \sum_{i=1}^k Z_{\frac{q}{q_i}} \times \cdots \times Z_{\frac{q}{q_k}} \pmod{q}.$$

(Można to udowodnić przez indukcję na k , stosując podstawowe chińskie twierdzenie o reszcie dla $k = 2$.) Ponadto poprzez rozszerzenie algorytmu w podrozdziale 8.1.5

²To nie przypadek, że nazwa algorytmu i czas jego działania są podobne do nazw ogólnego sító pól liczbowych do rozkładu na czynniki, ponieważ algorytmy mają wiele wspólnych kroków.

możliwa jest wydajna konwersja pomiędzy reprezentacją elementu jako elementu Zq i jego reprezentacją jako elementem $Zq_1 \times \dots \times Zq_k$.

Opisujemy teraz algorytm Pohliga – Hellmana. Mamy generator g i element h i chcemy znaleźć x takie, że $g = h$. Założymy, że znana jest faktoryzacja $i=1$ do k parami względnie pierwszą. (To nie musi być $q = p$ pełne rozłożenie na czynniki pierwsze q_j). Wiemy o tym

$$g^{q/q_i} \equiv q_i \pmod{q} = (g \pmod{q})^{\frac{1}{q_i}} = h \pmod{q_i} \text{ dla } i = 1, \dots, k. \quad (9.2)$$

Zagadnienie def q/q_i i $h \pmod{q_i}$ $\stackrel{\text{def}}{=} \text{godz.}/q_i$, mamy zatem k przypadków dyskretnego logarytmu $g \pmod{q_i}$ w k mniejszych grupach. W szczególności każdy problem $g = h \pmod{q_i}$ należy do podgrupy o rozmiarze $\text{ord}(g) = q_i$ (według lematu 9.4). (Zauważ, że każdy taki problem wyznacza jedynie $[x \pmod{q_i}]$; wynika to z Twierdzenia 8.53.)

Możemy rozwiązać każdy z k wynikowych przypadków, używając dowolnego algorytmu rozwiązywania problemu logarytmu dyskretnego. Rozwiążanie tych przypadków daje zbiór odpowiedzi $\{x_i\}_{i=1}^k$ dla których $g \pmod{q_i} = h \pmod{q_i}$. Twierdzenie 8.53 implikuje, że $x = x_i \pmod{q_i}$ dla wszystkich i . Ograniczenia wynikają z uogólnionego chińskiego twierdzenia o resztach omówionego wcześniej

$$x = x_1 \pmod{q_1}$$

$$\vdots$$

$$x = x_k \pmod{q_k}$$

jednoznacznie określić x modulo q , a pożądane rozwiązanie x można skutecznie zrekonstruować z $\{x_i\}$.

Przykład 9.5

Rozważ problem obliczania logarytmów dyskretnych w Z_{30} grupa 31, rzząd $q = 30 = 5 \cdot 3 \cdot 2$. Powiedzmy $g = 3$ i $h = 26 = g \cdot x$ z nieznanym x . Mamy: $(g^{30/5}) \pmod{30} = h$

$$30/5 (g^{30/3}) \pmod{30} = h \quad (36) \quad x = 266 \quad 16x = 1 \quad (310) \quad x$$

$$30/3 (g^{30/2}) \pmod{30} = h \quad = 2610 \quad 25x = 5 \quad (315) \quad x = 2615$$

$$30/2 \quad 30x = 30.$$

(Wszystkie powyższe równania są modulo 31.) Mamy $\text{ord}(16) = 5$, $\text{ord}(25) = 3$ i $\text{ord}(30) = 2$. Rozwiążując każde równanie, otrzymujemy

$$x = 0 \pmod{5}, x = 2 \pmod{3} \text{ i } x = 1 \pmod{2},$$

i tak $x = 5 \pmod{30}$. Rzeczywiście, $35 = 26 \pmod{30}$.

Jeśli q ma (znana) rozkład na czynniki pierwsze $q = \prod_{i=1}^k p_i^{e_i}$ następnie, stosując algorytm Pohliga-Hellmana, czas obliczenia logarytmów dyskretnych w grupie rzędu q jest zdominowany przez obliczenie logarytmu dyskretnego w podgrupie o rozmiarze $\max\{p_i^{e_i}\}$. Można to dalej zredukować do obliczenia dyskretnego podgrupie o rozmiarze $\max\{p_i\}$; patrz ćwiczenie 9.5.

9.2.2 Algorytm Baby-Step/Giant-Step

Algorytm małego kroku/gigantycznego kroku oblicza logarytmy dyskretne w grupie rzędu q za pomocą operacji grupowych $O(\sqrt{q})$. Pomyśl jest prosty. Mając generator $g \in G$, możemy sobie wyobrazić, że potęgi g tworzą cykl

$$1 = g^0, g^1, g^2, \dots, g^{q-2}, g^{q-1}, g^q = 1.$$

Wiemy, że h musi leżeć gdzieś w tym cyklu. Obliczenie wszystkich punktów w tym cyklu zajęłoby czas $\Omega(q)$. Zamiast tego „zaznaczamy” cykl w odstępach $= \sqrt{q}$; dokładniej, obliczamy zdecydowanie w rozmaitych przechowujemy $q/t+1 = O(\sqrt{q})$ elementy

$$g^0, g^t, g^{2t}, \dots, g^{q/t \cdot t}.$$

(Są to „gigantyczne kroki”). Zwróć uwagę, że przerwa między kolejnymi „znakami” wynosi co najwyżej t . Ponadto wiemy, że $h = g^x$ leży w jednej z tych luk. Zatem, jeśli następnie podejmiemy małe kroki i obliczymy t elementów

$$h \cdot g^1, \dots, h \cdot g^{t^2},$$

z których każda odpowiada „przesunięciu” h , wiemy, że jedna z tych wartości będzie równa jednemu z zaznaczonych przez nas punktów. Założymy, że znajdujemy $h \cdot g^i$. Możemy $h \cdot g^i = g^{kt+i}$. Wtedy łatwo obliczyć $\log g h := [(kt + i) \bmod q]$. Poniżej znajduje się pseudokod tego algorytmu.

ALGORYTM 9.6 Algorytm małego kroku/gigantycznego kroku

Dane wejściowe: Elementy $g, h \in G$; rzząd $q \in \mathbb{Z}$

Wynik: $\log g h$:=

- q dla $i = 0$ do q/t :
- oblicz $g^i := g^{i \cdot t}$
- posortuj pary (i, g^i) według ich drugiej składowej dla $i = 1$ do t :
- oblicz $h \cdot g^i := h \cdot g^{i \cdot t}$
- $h \cdot g^i = g^{kt+i}$ dla jakiegoś k , zwróć $[kt + i] \bmod q$

Algorytm wymaga $O(\sqrt{q})$ potęgowania/mnożeń w G . (W rzeczywistości, poza pierwszą wartością $g^1 = g$, każdą wartość g^i można obliczyć za pomocą pojedynczego mnożenia jako $g^i := g^{i-1} \cdot g^1$. Podobnie każde $h \cdot g^i$ można obliczyć jako $h \cdot g^{i-1} \cdot g^1$.) Sortowanie par $O(\sqrt{q}) \cdot \{\{i, g^i\}\}$ zajmuje czas $O(\sqrt{q} \cdot \log q)$, a następnie możemy użyć wyszukiwania binarnego, aby sprawdzić, czy każde $h \cdot g^i$ jest równe pewnym g^k w czasie $O(\log q)$. Ogólny algorytm działa zatem w czasie $O(\sqrt{q} \cdot \text{polilog}(q))$.

Przykład 9.7

Pokazujemy zastosowanie algorytmu w grupie cyklicznej $Z_q = \{1, 2, \dots, q-1\}$ porządku $q-1 = 28$. Weźmy $g = 2$ i $h = 17$. Ustawiamy $t = 5$ i obliczamy:

$$0_2 = 1, 2^5 = 3, 2^{10} = 9, 2^{15} = 27, 2^{20} = 23, 2^{25} = 11.$$

(Należy rozumieć, że wszystkie operacje znajdują się w Z_{29} .) Następnie oblicz:

$$17 \cdot 2^1 = 5, 17 \cdot 2^2 = 10, 17 \cdot 2^3 = 20, 17 \cdot 2^4 = 11,$$

4 i zauważ, że $17 \cdot 2^4 = 11 = 225$. Mamy zatem $\log_2 17 = 25 - 4 = 21$.

9.2.3 Logarytmy dyskretne ze zderzeń

Wadą algorytmu małego kroku/gigantycznego kroku jest to, że wykorzystuje on dużą ilość pamięci, ponieważ wymaga przechowywania punktów $O(q)$. Możemy otrzymać algorytm korzystający ze stałej pamięci – i mający ten sam asymptotyczny czas działania – poprzez wykorzystanie połączenia pomiędzy problemem logarytmu dyskretnego i hashowaniem odpornym na kolizje pokazanym w podrozdziale 8.4.2 i przywołanie ataku urodzinowego na małą przestrzeń w celu znalezienia kolizji z rozdziału 5.4.2.

Opisujemy pomysł na wysokim poziomie. Ustal podstawę $g \in G$ i jakiś element $h \in g$. Przypomnijmy, z wyników rozdziału 8.4.2, że jeśli zdefiniujemy funkcję skrótu $H_{g,h} : Z_q \times Z_q \rightarrow G$ przez $H_{g,h}(x_1, x_2) = g$, to znalezienie kolizji w $H_{g,h}$ dla $x_1 \neq x_2$ implikuje możliwość obliczyć $\log_g h$. (Patrz dowód Twierdzenia 8.79.) W ten sposób zredukowaliśmy problem obliczenia $\log_g h$ do znalezienia kolizji w funkcji mieszającej, coś, co wiemy, jak zrobić w czasie $O(|G|) = O(q)$ używając urodzinowego ataku! Co więcej, atak urodzinowy na małą przestrzeń spowoduje kolizję w tym samym czasie i stałej przestrzeni.

Pozostaje tylko zająć się kilkoma szczegółami technicznymi. Po pierwsze, atak urodzinowy na małą przestrzeń opisany w sekcji 5.4.2 zakłada, że zakres funkcji mieszającej jest podzbiorem jej domeny; w tym przypadku tak nie jest i w rzeczywistości (w zależności od reprezentacji zastosowanej dla elementów G) może się nawet zdarzyć, że $H_{g,h}$ nie ulega kompresji. Drugą kwestią jest to, że analiza w podrozdziale 5.4.2 potraktowała funkcję mieszającą jako funkcję losową, podczas gdy $H_{g,h}$ ma znaczącą część struktury algebraicznej.

Algorytm rho Pollarda zapewnia jeden ze sposobów radzenia sobie z powyższymi problemami. Opisujemy inny algorytm, który można postrzegać jako bardziej bezpośrednią realizację powyższych pomysłów. (W praktyce algorytm Pollarda byłby bardziej wydajny, chociaż oba algorytmy wykorzystują tylko operacje na grupach $O(q)$.) Niech $F : G \rightarrow Z_q \times Z_q$ oznaczają kryptograficzną funkcję skrótu uzyskaną np. przez odpowiednią modyfikację SHA-1. Zdefiniuj $H : G \rightarrow G$ przez $H(k) = H_{g,h}(F(k))$. Możemy użyć Algorytmu 5.9, z naturalnymi modyfikacjami, aby znaleźć kolizję w H przy użyciu ocen H w oczekiwaniu (i stałej pamięci) $O(|G|) = O(q)$. Z ogromnym prawdopodobieństwem prowadzi to do kolizji w $H_{g,h}$. Jesteś proszony o doprecyzowanie szczegółów w ćwiczeniu 9.6.

Warto tutaj zauważyc, że dowód bezpieczeństwa oparty na trudności problemu logarytmu dyskretnego —a mianowicie, że implikuje odporną na kolizje funkcję skrótu —prowadzi do lepszego algorytmu rozwiązania tego samego problemu! Mała refleksja powinna nas przekonać, że nie jest to zaskakujące: dowód poprzez redukcję pokazuje, że atak na jakąś konstrukcję (w tym przypadku znalezienie kolizji w funkcji skrótu) bezpośrednio skutkuje atakiem na podstawowe założenie (w tym przypadku twardość problemu logarytmu dyskretnego), co jest dokładnie tą właściwością, którą wykorzystuje powyższy algorytm.

9.2.4 Algorytm rachunku indeksowego

Zakończymy krótkim spojrzeniem na (nieogólny) algorytm rachunku indeksowego do obliczania logarytmów dyskretnych w grupie cyklicznej Z (dla p prime). W przeciwieństwie do poprzednich (ogólnych) algorytmów, to podejście charakteryzuje się czasem działania niższym niż wykładniczy. Algorytm ten przypomina w pewnym stopniu algorytm sita kwadratowego przedstawiony w podrozdziale 9.1.3 i zakładamy, że czytelnicy znają dyskusję tam zawartą. Podobnie jak w tym przypadku, omawiamy główne idee metody rachunku indeksowego, ale szczegółową analizę pozostawiamy poza zakresem naszego opracowania. Wprowadzono także pewne uproszczenia w celu przejrzystości prezentacji.

Metoda rachunku indeksowego wykorzystuje proces dwuetapowy. Co ważne, pierwszy krok wymaga znajomości jedynie modułu p i podstawy g , dlatego można go wykonać jako etap przetwarzania wstępnego, zanim znana będzie h —wartość, której logarytm dyskretny chcemy obliczyć. Z tego samego powodu wystarczy wykonać pierwszy krok tylko raz, aby rozwiązać wiele wystąpień problemu logarytmu dyskretnego (o ile wszystkie te wystąpienia mają te same p i g).

Krok 1. Napraw powiązane B i pozwól $\{p_1, \dots, p_k\}$ będzie zbiorem liczb pierwszych mniejszych lub równych B . W tym kroku znajdujemy x_1, \dots, x_{p-1} dla którego $g^{x_i} \equiv h \pmod{p}$ jest B -gładkie. Dokonuje się tego poprzez prostu wybiera uniform $\{x_i\}$, aż zostaną znalezione odpowiednie wartości.

Rozkładając otrzymane liczby B -gładkie, otrzymujemy równania:

$$\begin{aligned} g^{x_1} &= p_1^{e_{1,1}} \cdots p_k^{e_{1,k}} \pmod{p} \\ &\vdots \\ G^x &= p_1^{e_{x,1}} \cdots p_k^{e_{x,k}} \pmod{p} \end{aligned}$$

Biorąc logarytmy dyskretne, możemy przekształcić je w równania liniowe

$$x_1 = \sum_{j=1}^k e_{1,j} \cdot \logg{p_i} \pmod{p-1} \quad (9.3)$$

$$\vdots$$

$$x = \sum_{j=1}^k e_{i,j} \cdot \logg{p_i} \pmod{p-1}. \quad (9.4)$$

Należy zauważyć, że $\{x_i\}$ i $\{e_{i,j}\}$ są znane, natomiast $\{\logg{p_i}\}$ są nieznane.

Krok 2. Teraz mamy element h i chcemy obliczyć \logg{h} . Tutaj znajdujemy wartość x Zp^{-1} , dla której $[g \cdot x \cdot h \pmod{p}]$ jest B-gładkie.(Po raz kolejny można to zrobić po prostu wybierając równomierne x .) Powiedzmy

$$G^x \cdot \text{godz} = \sum_{j=1}^k p_i^{e_{i,j}} \pmod{\text{str}}$$

$$x + \logg{h} = \sum_{j=1}^k e_{i,j} \cdot \logg{p_i} \pmod{p-1},$$

gdzie znane są x i $\{e_{i,j}\}$. W połączeniu z równaniami (9.3)–(9.4) otrzymujemy $+1 \dots k+1$ równań liniowych z $k+1$ niewiadomymi $\{\logg{p_i}\}$ i \logg{h} . Stosując metody algebraiczne liniowe³ (przy założeniu, że układ równań nie jest niedookreślony) możemy rozwiązać każdą z niewiadomych, a w szczególności otrzymać pożądane rozwiązanie \logg{h} .

Przykład 9.8

Niech $p = 101$, $g = 3$ i $h = 87$. Mamy $[310 \pmod{101}] = 65 = 5 \cdot 13$.

Podobnie $[312 \pmod{101}] = 80 = 24 \cdot 5$ i $[314 \pmod{101}] = 13$. Mamy zatem równania liniowe

$$10 = \log_3 5 + \log_3 13 \pmod{100} \\ 4 \cdot \log_3 2 + \log_3 5 \pmod{100} \\ 14 = \log_3 13 \pmod{100}.$$

Mamy również $35 \cdot 87 = 32 = 25 \pmod{101}$, lub

$$5 + \log_3 87 = 5 \cdot \log_3 2 \pmod{100}. \quad (9.5)$$

Dodając drugie i trzecie równanie i odejmując pierwsze, otrzymujemy $4 \cdot \log_3 2 = 16 \pmod{100}$. Nie określa to jednoznacznie $\log_3 2$ (ponieważ 4 to

³Technicznie rzecz biorąc, wszystko jest nieco bardziej skomplikowane, ponieważ wszystkie równania liniowe są modulo $p-1$, co nie jest liczbą pierwszą. Istnieją jednak techniki radzenia sobie z tym problemem.

nie jest odwracalny modulo 100), ale mówi nam, że $\log_3 2 = 4, 29, 54$ lub 79 (por. ćwiczenie 9.3). Wypróbowanie wszystkich możliwości daje $\log_3 2 = 29$. Podłączenie tego do Równanie (9.5) daje $\log_3 87 = 40$.

Czas pracy. Wybór większej wartości B zwiększa prawdopodobieństwo, że jednolita wartość w Z_p jest B-gładki; oznacza to jednak, że będziemy musieli częściej pracować zidentyfikować i rozłożyć na czynniki liczby B-gładkie, a będziemy musieli znaleźć ich więcej ich. Ponieważ układ równań będzie większy, rozwiązywanie układu będzie większe zająć dłużej. Wybór optymalnej wartości B daje algorytm, który (czas heuris-in $2O(\log p \log \log p)$ przynajmniej merytorycznie) oblicza logarytmy dyskretnie w Z_p . (W rzeczywistości stały składnik wykładnika można określić dość precyzyjnie.) Dla naszych celów ważne jest to, że jest to zjawisko subwykładnicze w długość str.

9.3 Zalecane długości kluczy

Znajomość najlepszych dostępnych algorytmów rozwiązywania różnych problemów kryptograficznych problemów jest niezbędne do określenia odpowiedniej długości klucza, którą należy osiągnąć pewien pożądany poziom bezpieczeństwa. W poniższej tabeli podsumowano długości kluczy obecnie rekomendowane przez amerykański Narodowy Instytut Standardów i Technologii⁴ (NIST) [13]:

Skuteczny Długość modułu długości klucza	RSA	Logarytm dyskretny	
		Zamówienie-q Podgrupa Z_p	Krzywa eliptyczna Zamówienie grupowe q
112	2048	p: 2048, q: 224 p:	224
128	3072	3072, q: 256 p:	256
192	7680	7680, q: 384 p:	384
256	15360	15360, q: 512	512

Wszystkie wartości w tabeli są mierzone w bitach. „Efektywna długość klucza” wynosi wartość w taką, jaką przyjmuje najbardziej znany algorytm rozwiązania problemu czas około $2n$; tj. trudność obliczeniowa rozwiązania problemu wynosi w przybliżeniu równoważne przeszukiwaniu metodą brute-force schemat klucza symetrycznego z kluczem n-bitowym lub czas znalezienia kolizji w funkcja skrótu o długości wyjściowej $2n$ bitów. NIST uważa, że klucz efektywny ma 112 bitów długość akceptowalna ze względów bezpieczeństwa do roku 2030, ale zalecana jest wersja 128-bitowa lub większe długości kluczy do zastosowań, w których bezpieczeństwo wymagane jest później.

⁴Inne grupy przedstawiły własne zalecenia; zobacz <http://keylength.com>.

Biorąc pod uwagę to, czego dowiedzieliśmy się w tym rozdziale, pouczające jest przyjrzenie się bliżej niektórym liczbom w tabeli. Pierwszą rzeczą, którą możemy zauważyc, jest to, że grupy krzywych eliptycznych można wykorzystać do realizacji dowolnego poziomu bezpieczeństwa przy mniejszych parametrach niż w przypadku . Dzieje się tak po prostu dlatego RSA lub podgrup Z. Nie są znane żadne algorytmy podwykładowicze do rozwiązywania problemu logarytmu dyskretnego w takich grupach (przy odpowiednim wyborze). Jednakże osiągnięcie n-bitowego bezpieczeństwa wymaga grupy o krzywej eliptycznej, której rząd q ma długość $2n$ bitów. Jest to konsekwencja ogólnych algorytmów, które widzieliśmy w tym rozdziale, które rozwiązują problem logarytmu dyskretnego (w dowolnej grupie) w czasie $O(\sqrt{q})$.

Wracając do sprawy Z_p widzimy, że tutaj również potrzebna jest $2n$ -bitowa wartość q dla n-bitowego bezpieczeństwa. Długość p musi być jednak znacznie większa, ponieważ sítu pól liczbowych można wykorzystać do obliczenia logarytmów dyskretnych w czasie

p podwykładowiczym na długości p . (Oznacza to, że $p \equiv q$ wybiera się Z w taki sposób, że czas działania sítu pól liczbowych, który zależy od długości p , i czas działania algorytmu ogólnego, który zależy od długości q , będą w przybliżeniu równe.) Praktyczne konsekwencje tego są takie, że dla dowolnego pożądanego poziomu bezpieczeństwa kryptosystemy o krzywej eliptycznej mogą wykorzystywać znacznie mniejsze parametry, z asymptotycznie szybszymi operacjami grupowymi dla uczciwych stron, niż kryptosystemy oparte na podgrupach Z_p .

Referencje i dodatkowe lektury

Algorytm p-1 Pollarda został opublikowany w 1974 r. [139], a jego metodę rho dla faktoringu opisano w roku następnym [140]. Algorytm sítu kwadratowego jest dzielem Pomerance'a [142], bazującego na wcześniejszych pomysłach Dixon [60].

Algorytm małego kroku/gigantycznego kroku został opracowany przez Shanksa [153]. Algorytm Pohliga-Hellmana został opublikowany w 1978 r. [138], podobnie jak algorytm rho Pollarda do obliczania logarytmów dyskretnych [141]. Algorytm rachunku indeksowego, jak go opisaliśmy, jest autorstwa Adlemana [4].

Teksty Wagstaffa [173], Shoupa [159], Crandalla i Pomerance'a [51], Joux [96] i Galbraitha [69] dostarczają dalszych informacji na temat algorytmów rozkładu na czynniki i obliczania logarytmów dyskretnych, w tym opisy (gen -eral) sítu polowe numeryczne. Całkiem niedawno ogłoszono ulepszony algorytm problemu logarytmu dyskretnego w polach skończonych o małej charakterystyce [11].

Dolne granice czasu działania algorytmów generycznych do obliczania logarytmów dyskretnych, które asymptotycznie odpowiadają czasom działania opisanych tutaj algorytmów, podali Nechaev [133] i Shoup [155].

Lenstra i Verheul [113] obszerne omawiają, w jaki sposób znane algorytmy rozkładu na czynniki i obliczania logarytmów dyskretnych wpływają na dobór parametrów kryptograficznych w praktyce.

Ćwiczenia

9.1 Aby przyspieszyć algorytm generowania klucza dla RSA, zasugerowano wygenerowanie liczby pierwszej poprzez wygenerowanie wielu małych losowych liczb pierwszych, pomnożenie ich przez siebie i dodanie (oczywiście po sprawdzeniu, czy wynik jest liczbą pierwszą). Pomijając kwestię prawdopodobieństwa, że taka wartość rzeczywiście jest liczbą pierwszą, co sądzisz o tej metodzie?

W wykonaniu algorytmu 9.2, zdefiniuj x w danym $\stackrel{(I)}{=} F(i)(x(0))$. Pokaż, że jeśli, 9.2 wykonaniu algorytmu, istnieje $i, j \in 2^{n/2}$ takie, że $(j) \bmod p$, to wykonanie algorytmu $x(I) = x(j)$ ale $x(i) = x$ daje p . (Analiza różni się nieco od analizy algorytmu 5.9, ponieważ algorytmy —i ich cele —są nieco inne.)

9.3 (a) Pokaż, że jeśli $ab = c \bmod N$ i $\gcd(b, N) = d$, to: i. $d \mid C$; II. $a \cdot (b/d)$

$$\begin{aligned} &= (c/d) \\ &\bmod (N/d); \text{ i iii. } \gcd(b/d, N/d) = 1. \end{aligned}$$

(b) Opisz, jak wykorzystać powyższe do wydajnego obliczania logarytmów dyskretnych w Z_N , nawet jeśli podstawa g nie jest generatorem Z_N .

9.4 Tutaj pokazujemy, jak rozwiązać problem logarytmu dyskretnego w czasie cyklicznym rzędu $q = p$ erator $g \stackrel{\text{mi}}{=} O(\text{polylog}(q) \cdot p)$. Biorąc pod uwagę grupę genów rzędu $q = p$ i wartość h , chcemy obliczyć $x = \log g h$.

(a) Pokaż, jak obliczyć $[x \bmod p]$ w czasie $O(\text{polylog}(q) \cdot p)$.

Wskazówka: rozwiąż równanie

$$\underset{\text{sol}}{x_0} \stackrel{e-1}{\cdot} \underset{\text{sol}}{p} = \text{godz}^{e-1}$$

i użyj tych samych pomysłów, co w algorytmie Pohliga – Hellmana.

(b) Powiedzmy, że $x = x_0 + x_1 \cdot p + \dots + x_{e-1} \cdot p^{e-1}$ gdzie $0 \leq x_i < p$. W poprzednim kroku określiliśmy x_0 . Pokaż, jak obliczyć w czasie $\text{polilog}(q)$ wartość h^1 taką, że $(g^p)^{x_1 + x_2 \cdot p + \dots + x_{e-2} \cdot p^{e-2}} = h^1$.

(c) Użyj rekurencji, aby uzyskać deklarowany czas działania oryginału problem. (Zauważ, że $e = O(\log q)$.)

9.5 Niech q będzie rozkładem na czynniki pierwsze $q = \prod_{i=1}^k p_i^{\alpha_i}$. Korzystając z wyniku z poprzedniego zadania, pokaż modyfikację algorytmu Pohliga–Hellmana rozwiązującą problem logarytmu dyskretnego w grupie rzędu q w czasie $O(\text{polilog}(q) \cdot \sum_{i=1}^k \alpha_i)$.

9.6 Podaj pseudokod algorytmu małej przestrzeni do obliczania logarytmów dyskretnych opisanego w podrozdziale 9.2.3 i podaj heurystyczną analizę prawdopodobieństwa jego powodzenia.

Rozdział 10

Zarządzanie kluczami i Rewolucja klucza publicznego

10.1 Dystrybucja kluczy i zarządzanie kluczami

W rozdziałach 1–7 widzieliśmy, jak można wykorzystać kryptografię klucza prywatnego, aby zapewnić poufność i integralność dwóm stronom komunikującym się za pośrednictwem niezabezpieczonego kanału, zakładając, że obie strony posiadają wspólny, tajny klucz. Jednakże pytanie, które odłożyliśmy od rozdziału 1, brzmi:

W jaki sposób strony mogą w ogóle dzielić się tajnym kluczem?

Jest oczywiste, że klucza nie można po prostu przesłać publicznym kanałem komunikacji, ponieważ podsłuchujący przeciwnik mógłby go wtedy obserwować po drodze. Zamiast tego należy zastosować jakiś inny mechanizm.

W niektórych sytuacjach strony mogą mieć dostęp do bezpiecznego kanału, za pomocą którego mogą niezawodnie udostępnić tajny klucz. Typowym przykładem jest sytuacja, w której obie strony znajdują się w tym samym momencie i mogą dzielić klucz. Alternatywnie strony mogą skorzystać z zaufanej usługi kurierskiej jako bezpiecznego kanału. Podkreślamy, że fakt, że strony mogą dzielić klucz – a zatem muszą w pewnym momencie mieć dostęp do bezpiecznego kanału – nie oznacza, że kryptografia klucza prywatnego jest bezużyteczna: w pierwszym przykładzie strony mają w pewnym momencie bezpieczny kanał czas, ale nie na czas nieokreślony; w drugim przykładzie korzystanie z bezpiecznego kanału może być wolniejsze i bardziej kosztowne niż komunikacja przez niezabezpieczony kanał.

Powyższe podejścia zostały wykorzystane do dzielenia się kluczami w kontekście rządowym, dyplomatycznym i wojskowym. Na przykład „czarny telefon” łączący Moskwę z Waszyngtonem w latach 60. XX wieku był szyfrowany za pomocą jednorazowej podkładki, której klucze dzielili kurierzy, którzy latali z jednego kraju do drugiego z teczkami pełnymi wydruków. Takie podejście można zastosować także w korporacjach, np. do ustalenia wspólnego klucza pomiędzy centralną bazą danych a nowym pracownikiem pierwszego dnia pracy. (Wróćmy do tego przykładu w następnej sekcji.)

Jednak poleganie na bezpiecznym kanale dystrybucji kluczy nie sprawdza się dobrze w wielu innych sytuacjach. Rozważmy na przykład dużą, międzynarodową korporację, w której każda para pracowników może potrzebować możliwości bezpiecznej komunikacji, chronionej również przed innymi pracownikami.

Delikatnie mówiąc, niewygodne będzie spotkanie się każdej pary pracowników w celu bezpiecznego udostępnienia klucza; dla pracowników pracujących w różnych miastach może to być nawet niemożliwe. Nawet jeśli aktualna grupa pracowników mogłaby w jakiś sposób dzielić się kluczami między sobą, udostępnianie im kluczy nowym pracownikom, którzy dołączą po zakończeniu wstępniego udostępniania, byłoby niepraktyczne.

Zakładając, że tych N pracowników jest w stanie w jakiś sposób bezpiecznie dzielić się ze sobą kluczami, kolejną istotną wadą jest to, że każdy pracownik będzie musiał zarządzać i przechowywać N - 1 tajnych kluczy (po jednym dla każdego innego pracownika). W rzeczywistości może to znacznie zniżyć liczbę kluczy przechowywanych przez każdego użytkownika, ponieważ pracownicy mogą również potrzebować kluczy do bezpiecznej komunikacji ze zdalnymi zasobami, takimi jak bazy danych, serwery, drukarki itd.

Rozpowszechnianie tak wielu tajnych kluczy stanowi poważny problem logistyczny. Co więcej, wszystkie te klucze muszą być bezpiecznie przechowywane. Im więcej kluczy, tym trudniej je chronić i tym większe jest ryzyko kradzieży niektórych kluczy przez atakującego.

Systemy komputerowe są często infekowane przez wirusy, robaki i inne formy złośliwego oprogramowania, które mogą kraść tajne klucze i wysyłać je po cichu przez sieć do atakującego. Tym samym przechowywanie kluczy na komputerach osobistych pracowników nie zawsze jest bezpiecznym rozwiązaniem.

Żeby było jasne, potencjalne naruszenie tajnych kluczy zawsze stanowi problem, niezależnie od liczby kluczy posiadanych przez każdą ze stron. Jeśli jednak trzeba przechowywać tylko kilka kluczy, dostępne są dobre rozwiązania pozwalające uporać się z tym zagrożeniem. Typowym rozwiązaniem jest obecnie przechowywanie kluczy na bezpiecznym sprzęcie, takim jak karta inteligentna. Karta inteligentna może przeprowadzać obliczenia kryptograficzne przy użyciu przechowywanych tajnych kluczy, zapewniając, że klucze te nigdy nie trafią na komputery osobiste użytkowników. Prawidłowo zaprojektowana karta inteligentna może być znacznie bardziej odporna na ataki niż komputer osobisty —na przykład zazwyczaj nie może zostać zainfekowana złośliwym oprogramowaniem —dzięki czemu stanowi dobry sposób ochrony tajnych kluczy użytkowników. Niestety, karty inteligentne mają zazwyczaj dość ograniczoną pamięć i dlatego nie mogą przechowywać setek (lub tysięcy) kluczy.

Wszystkie problemy opisane powyżej można rozwiązać – w zasadzie, nawet jeśli nie w praktyce – w „zamkniętych” organizacjach składających się z dobrze określonej populacji użytkowników, z których wszyscy są skłonni przestrzegać tych samych zasad dystrybucji i przechowywania kluczy. Załamują się jednak w „systemach otwartych”, w których użytkownicy wchodzą w interakcje przejściowe, nie mogą umówić się na fizyczne spotkanie i mogą nawet nie być świadomi swojego istnienia aż do momentu, w którym chcą się porozumieć. W rzeczywistości jest to sytuacja bardziej powszechna, niż mogłoby się wydawać na pierwszy rzut oka: rozwija się użycie szyfrowania w celu wysłania informacji o karcie kredytowej do sprzedawcy internetowego, od którego niczego wcześniej nie kupiłeś, lub wysłania wiadomości e-mail do osoby, której nigdy wcześniej nie kupowałeś. spotkałem się osobiście. W takich przypadkach sama kryptografia klucza prywatnego po prostu nie zapewnia rozwiązania i musimy szukać dalej odpowiednich rozwiązań.

Podsumowując, istnieją co najmniej trzy różne problemy związane ze stosowaniem kryptografii klucza prywatnego. Pierwsza dotyczy dystrybucji kluczy; drugi polega na przechowywaniu i zarządzaniu dużą liczbą tajnych kluczy; trzecim jest brak możliwości zastosowania kryptografii klucza prywatnego w systemach otwartych.

10.2 Rozwiążanie częściowe: Centra dystrybucji kluczy

Jednym ze sposobów rozwiązywania niektórych problemów z poprzedniej sekcji jest użycie centrum dystrybucji kluczy (KDC) w celu ustanowienia kluczy współdzielonych. Rozważmy jeszcze raz przypadek dużej korporacji, w której wszystkie pary pracowników muszą mieć możliwość bezpiecznej komunikacji. W takiej sytuacji możemy wykorzystać fakt, że wszyscy pracownicy mogą ufać jakiemuś podmiotowi – powiedzmy administratorowi systemu – przynajmniej w kwestii bezpieczeństwa informacji związanych z pracą. Ta zaufana jednostka może następnie działać jako KDC i pomagać wszystkim pracownikom w udostępnianiu kluczy parami.

Kiedy dołączy nowy pracownik, KDC może udostępnić mu klucz (osobiście, w bezpiecznym miejscu) w ramach pierwszego dnia pracy tego pracownika.

Jednocześnie KDC mogłoby również dystrybuować klucze współdzielone pomiędzy tym pracownikiem a wszystkimi obecnymi pracownikami. Oznacza to, że kiedy i-ty pracownik dołączy, KDC może (oprócz dzielenia się kluczem między sobą a tym nowym pracownikiem) wygenerować i-1 klucze k_1, \dots, k_{i-1} , przekaż te klucze nowemu pracownikowi, a następnie wyślij klucz k_j -temu istniejącemu pracownikowi, szyfrując go kluczem, który pracownik już dzieli z KDC. Następnie nowy pracownik dzieli klucz z każdym innym pracownikiem (a także z KDC).

Lepszym podejściem, które pozwala uniknąć wymagania od pracowników przechowywania wielu kluczy i zarządzania nimi, jest wykorzystanie KDC w trybie online do generowania kluczy „na żądanie”, ilekroć dwóch pracowników chce się bezpiecznie komunikować. Tak jak poprzednio, KDC udostępnii (inny) klucz każdemu pracownikowi, co można bezpiecznie zrobić pierwszego dnia pracy pracownika. Założmy, że KDC dzieli klucz k_A z pracownikiem Alice i k_B z pracownikiem Bobem. Później, gdy Alicja będzie chciała bezpiecznie komunikować się z Bobem, może po prostu wysłać wiadomość „Ja, Alicja, chcę porozmawiać z Bobem” do KDC. (W razie potrzeby wiadomość ta może zostać uwierzytelniona przy użyciu klucza udostępnionego Alicji i KDC.)

Następnie KDC wybiera nowy, losowy klucz —zwany kluczem sesji —i wysyła ten klucz k do Alicji zaszyfrowanej przy użyciu k_A i do Boba zaszyfrowanego przy użyciu k_B .

(Ten protokół jest zbyt uproszczony, aby można go było zastosować w praktyce; dalsze omówienie poniżej). Kiedy Alicia i Bob odzyskają ten klucz sesji, będą mogli go używać do bezpiecznej komunikacji. Kiedy skończą rozmowę, mogą (i powinni) usunąć klucz sesji, ponieważ zawsze mogą ponownie skontaktować się z KDC, jeśli chcą się porozumieć później.

Rozważ zalety tego podejścia:

1. Każdy pracownik musi przechowywać tylko jeden długoterminowy tajny klucz (czyli ten, który udostępnia KDC). Pracownicy nadal muszą zarządzać kluczami sesji i je przechowywać, ale są to klucze krótkoterminowe, które są usuwane po zakończeniu sesji komunikacyjnej.

Centrum dystrybucji kluczy musi przechowywać wiele kluczy długoterminowych. Jednakże KDC można przechowywać w bezpiecznym miejscu i zapewnić najwyższą możliwą ochronę przed atakami sieciowymi.

2. Kiedy pracownik przyłącza się do organizacji, jedyne co należy zrobić to ustawić klucz pomiędzy tym pracownikiem a KDC. Żaden inny pracownik nie musi aktualizować posiadanego zestawu kluczy.

W ten sposób centra KDC mogą złagodzić dwa problemy, które zaobserwowałyśmy w odniesieniu do kryptografii klucza prywatnego: mogą uprościć dystrybucję kluczy (ponieważ w przypadku dołączania pracownika musi być udostępniony tylko jeden nowy klucz i rozsądne jest założenie bezpiecznego kanału między KDC i tego pracownika pierwszego dnia pracy) i może zmniejszyć złożoność przechowywania kluczy (ponieważ każdy pracownik musi przechowywać tylko jeden klucz). Centra KDC wnoszą ogromny wkład w uczynienie kryptografii kluczem prywatnym praktycznym w dużych organizacjach, w których istnieje jeden podmiot, któremu wszyscy ufają.

Poleganie na KDC ma jednak pewne wady:

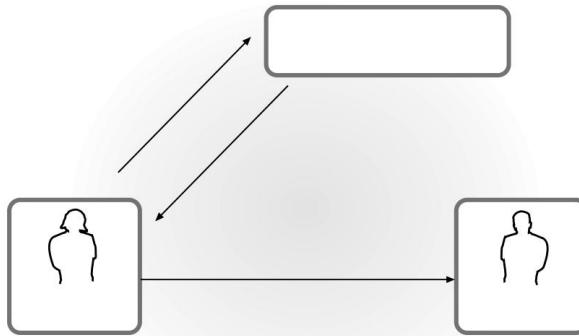
1. Udany atak na KDC spowoduje całkowite uszkodzenie systemu: osoba atakująca może złamać wszystkie klucze, a następnie podsłuchać cały ruch sieciowy. To sprawia, że KDC jest celem o dużej wartości. Należy pamiętać, że nawet jeśli KDC jest dobrze chronione przed atakami zewnętrznymi, zawsze istnieje możliwość ataków wewnętrznych ze strony pracowników mających dostęp do KDC (na przykład kierownika IT).
2. KDC to pojedynczy punkt awarii: w przypadku awarii KDC bezpieczna komunikacja jest tymczasowo niemożliwa. Jeśli pracownicy stale kontaktują się z KDC i proszą o ustalenie kluczy sesji, obciążenie KDC może być bardzo duże, zwiększaając w ten sposób ryzyko, że KDC nie będzie reagować lub będzie działać wolno.

Prostym rozwiązaniem drugiego problemu jest replikacja KDC. To działa (i jest stosowane w praktyce), ale oznacza też, że na system jest teraz więcej punktów ataku. Dodanie większej liczby centrów KDC utrudnia także dodawanie nowych pracowników, ponieważ aktualizacje muszą być bezpiecznie przesyłane do każdego centrum dystrybucji.

Protokoły dystrybucji kluczy przy użyciu KDC. W literaturze istnieje wiele protokołów bezpiecznej dystrybucji kluczy przy użyciu KDC. Wspominamy w szczególności o protokole Needhama – Schrödera, który stanowi rdzeń Kerberos, ważnej i szeroko stosowanej usługi służącej do przeprowadzania uwierzytelniania i wspierania bezpiecznej komunikacji. (Kerberos jest używany na wielu uniwersytetach i korporacjach i jest domyślnym mechanizmem obsługującym bezpieczne uwierzytelnianie sieciowe i komunikację w systemie Windows i wielu systemach UNIX.)

Zwracamy uwagę tylko na jedną cechę tego protokołu. Kiedy Alicja kontaktuje się z KDC i prosi o komunikację z Bobem, KDC nie wysyła zaszyfrowanego klucza sesji zarówno do Alicji, jak i Boba, jak opisaliśmy wcześniej. Zamiast tego KDC wysyła Alicji klucz sesji zaszyfrowany kluczem Alicji oprócz klucza sesji zaszyfrowanego kluczem Boba. Następnie Alicja przekazuje drugi zaszyfrowany tekst Bobowi, jak pokazano na rysunku 10.1. Drugi zaszyfrowany tekst jest czasami nazywany biletą i można go postrzegać jako dane uwierzytelniające, które pozwalają Alicji rozmawiać z Bobem (i dają Bobowi pewność, że rozmawia z Alicją).

Rzeczywiście,



RYSUNEK 10.1: Ogólny szablon protokołów dystrybucji kluczy.

choćież nie podkreśialiśmy tego punktu w naszej dyskusji, podejście oparte na KDC może również zapewnić użyteczny sposób przeprowadzania uwierzytelniania. Zauważ także, że Alicja i Bob nie muszą być jednocześnie użytkownikami; Alicja może być użytkownikiem, a Bob zasobem, takim jak serwer, dysk zdalny lub drukarka.

Protokół został zaprojektowany w ten sposób, aby zmniejszyć obciążenie KDC. W opisany protokole KDC nie musi inicjować drugiego połączenia z Bobem i nie musi się martwić, czy Bob jest on-line, gdy Alicja inicjuje protokół. Co więcej, jeśli Alicja zatrzyma bilet (i swoją kopię klucza sesji), może ponownie zainicjować bezpieczną komunikację z Bobem, po prostu ponownie wysyłając bilet do Boba, bez żadnego zaangażowania KDC. (W praktyce bilety wygasają i ostatecznie wymagają odnowienia. Sesję można jednak wznowić w akceptowalnym terminie).

Na zakończenie zauważamy, że w praktyce kluczem, który Alicja udostępnia KDC, może być krótkie, łatwe do zapamiętania hasło. W takim przypadku pojawia się wiele dodatkowych problemów związanych z bezpieczeństwem, którymi należy się zająć. Domyślnie zakładaliśmy również, że atakujący będzie tylko pasywnie podsłuchiwać, a nie taki, który mógłby aktywnie próbować ingerować w protokół. Zainteresowanego czytelnika odsyłamy do odnośników na końcu tego rozdziału, gdzie można znaleźć więcej informacji o tym, jak można rozwiązać takie problemy.

10.3 Wymiana kluczy i protokół Diffiego–Hellmana

W praktyce powszechnie stosuje się KDC i protokoły takie jak Kerberos. Jednak takie podejście do problemu dystrybucji kluczy w pewnym momencie nadal wymaga prywatnego i uwierzytelnionego kanału, którego można używać do udostępniania kluczy. (W szczególności założyliśmy istnienie takiego kanału pomiędzy KDC a pracownikami już pierwszego dnia pracy.) Zatem nadal nie mogą rozwiązać problemu

dystrybucji kluczy w systemach otwartych, takich jak Internet, gdzie może nie być dostępny kanał prywatny pomiędzy dwoma użytkownikami pragnącymi się komunikować.

Aby osiągnąć prywatną komunikację bez konieczności komunikowania się za pośrednictwem kanału prywatnego, potrzebne jest radykalnie odmienne podejście. W 1976 roku Whitfield Diffie i Martin Hellman opublikowali artykuł o niewinnie wyglądającym tytule „Nowe kierunki w kryptografii”. W pracy tej zaobserwowali, że na świecie często występuje asymetria; w szczególności istnieją pewne działania, które można łatwo wykonać, ale niełatwo je odwrócić. Na przykład kłódki można zamknąć bez klucza (tzn. łatwo), ale później nie można ich ponownie otworzyć. Co bardziej uderzające, szklany wazon łatwo jest rozbić, ale niezwykle trudno go ponownie złożyć. Algorytmicznie (co jest bardziej istotne dla naszych celów) łatwo jest pomnożyć dwie duże liczby pierwsze, ale trudno je odzyskać z ich iloczynu. (To jest dokładnie problem faktoringu omawiany w poprzednich rozdziałach). Diffie i Hellman zdali sobie sprawę, że takie zjawisko można wykorzystać do opracowania interaktywnych protokołów bezpiecznej wymiany kluczy, które umożliwiają dwóm stronom dzielenie się tajnym kluczem poprzez komunikację kanałem publicznym, poprzez strony wykonują operacje, które mogą odwrócić, ale których podsłuchujący nie może.

Istnienie bezpiecznych protokołów wymiany kluczy jest dość zdumiewające. Oznacza to, że ty i twój przyjaciel możecie zgodzić się na sekret, po prostu krzyżując przez pokój (i wykonując lokalne obliczenia); tajemnica nie byłaby znana nikomu innemu, nawet gdyby wysłuchała wszystkiego, co zostało powiedziane. Rzeczywiście, aż do 1976 roku powszechnie uważano, że nie da się osiągnąć bezpiecznej komunikacji bez uprzedniego udostępnienia tajnych informacji kanałem prywatnym.

Wpływ artykułu Diffiego i Hellmana był ogromny. Oprócz wprowadzenia zasadniczo nowego spojrzenia na kryptografię, był to jeden z pierwszych kroków w kierunku przeniesienia kryptografii z domeny prywatnej do domeny publicznej. Cytujemy dwa pierwsze akapity ich artykułu:

Stoimy dziś u progu rewolucji w kryptografii. Rozwój taniego sprzętu cyfrowego uwolnił go od ograniczeń konstrukcyjnych obliczeń mechanicznych i obniżył koszty wysokiej jakości urządzeń kryptograficznych do poziomu, w którym można je stosować w zastosowaniach komercyjnych, takich jak zdalne bankomaty i terminale komputerowe.

Z kolei tego typu zastosowania stwarzają zapotrzebowanie na nowe typy systemów kryptograficznych, które minimalizują konieczność stosowania bezpiecznych kanałów dystrybucji kluczy. . . Jednocześnie teoretyczny rozwój teorii informacji i informatyki daje nadzieję na dostarczenie bezpiecznych kryptosystemów, które można udowodnić, zmieniając tę starożytną sztukę w naukę.

Diffie i Hellman nie przesadzali, a rewolucja, o której mówili, była w dużej mierze efektem ich pracy.

W tej sekcji przedstawiamy protokół wymiany kluczy Diffiego-Hellmana. Udowodnimy jego bezpieczeństwo przed podsłuchiwaniem przez przeciwników lub, równoważnie, pod-

założenie, że strony komunikują się za pośrednictwem publicznego, ale uwierzytelnionego kanału (aby osoba atakująca nie mogła ingerować w ich komunikację). Bezpieczeństwo przed podsłuchującym przeciwnikiem jest stosunkowo słabą gwarancją i w praktyce protokoły wymiany kluczy muszą spełniać silniejsze koncepcje bezpieczeństwa, które wykraczają poza nasze obecne możliwości. (Co więcej, interesuje nas tutaj sytuacja, w której komunikujące się strony nie podzieliły się wcześniej informacjami, w takim przypadku nie można nic zrobić, aby zapobiec podszywaniu się przeciwnika pod jedną z stron. Powrócimy do tego punktu później).

Ustanawianie i definicja bezpieczeństwa. Rozważamy ustawienie z dwiema stronami – tradycyjnie zwanymi Alicją i Bobem – które uruchamiają protokół probabilistyczny Π w celu wygenerowania wspólnego, tajnego klucza; Π można postrzegać jako zbiór instrukcji dla Alicji i Boba w protokole. Alicja i Bob zaczynają od przytrzymania parametru bezpieczeństwa $1n$; następnie uruchamiają Π , używając (niezależnych) losowych bitów. Na koniec protokołu Alicja i Bob wyprowadzają odpowiednio klucze $k_A, k_B \in \{0, 1\}^n$. Podstawowym wymogiem poprawności jest to, że $k_A = k_B$. Ponieważ będziemy mieli do czynienia tylko z protokołami spełniającymi ten wymóg, będziemy mówić po prostu o kluczu $k = k_A = k_B$ wygenerowanym przez uczciwe wykonanie Π .

Przejdźmy teraz do definicji bezpieczeństwa. Intuicyjnie protokół wymiany kluczy jest bezpieczny, jeśli klucz wydany przez Alicję i Boba jest całkowicie nie do odgadnięcia przez podsłuchującego przeciwnika. Jest to formalnie zdefiniowane poprzez wymaganie, aby przeciwnik, który podsłuchał wykonanie protokołu, nie był w stanie odróżnić klucza k wygenerowanego w wyniku tego wykonania (i obecnie współdzielonego przez Alicję i Boba) od jednolitego klucza o długości n . Jest to znacznie silniejsze niż po prostu wymaganie, aby przeciwnik nie był w stanie dokładnie obliczyć k , a to silniejsze pojęcie jest konieczne, jeśli strony będą później używać k w niektórych zastosowaniach kryptograficznych (np. jako klucz do schematu szyfrowania klucza prywatnego).

Formalizując powyższe, niech Π będzie protokołem wymiany kluczy, A przeciwnikiem, oraz n parametrem bezpieczeństwa. Mamy następujący eksperyment:

Eksperyment wymiany klucza $KEav A, \Pi(n)$:

1. Dwie strony posiadające 1^n wykonują protokół Π . Powoduje to transkrypcję trans zawierającą wszystkie wiadomości wysłane przez strony i klucz k wyjściowy każdej ze stron.
2. Wybierany jest bit jednolity $b \in \{0, 1\}$. Jeśli $b = 0$, ustaw $\hat{k} := k$, a jeśli $b = 1$, to wybierz $\hat{k} \in \{0, 1\}^n$ równomiernie losowo.
3. A ma dane trans i \hat{k} i daje na wyjściu bit b .
4. Wynik eksperymentu definiuje się jako 1, jeśli $b = \hat{k}$, i 0 w przeciwnym razie. (W przypadku gdy $KEav A, \Pi(n) = 1$, mówimy, że A się powiedzie.)

A otrzymuje trans, aby uchwycić fakt, że A podsłuchuje całą realizację protokołu i tym samym widzi wszystkie wiadomości wymieniane przez strony. W prawdziwym świecie A nie otrzymałyby żadnego klucza; w eksperymencie przeciwnik otrzymuje \hat{k} jedynie w celu zdefiniowania, co dla A oznacza „złamanie” zabezpieczenia Π .

Oznacza to, że przeciwnikowi uda się „złamać” Π , jeśli potrafi poprawnie określić, czy klucz \hat{k} jest kluczem rzeczywistym odpowiadającym danemu wykonaniu protokołu, czy też \hat{k} jest kluczem jednolitym, niezależnym od transkrypcji.

Zgodnie z oczekiwaniami mówimy, że Π jest bezpieczne, jeśli przeciwnik odniesie sukces z prawdopodobieństwem co najwyżej pomijalnie większym niż $1/2$. To jest:

DEFINICJA 10.1 Protokół wymiany kluczy Π jest bezpieczny w obecności podsłuchiwacza, jeśli dla wszystkich probabilistycznych przeciwników A w czasie wielomianowym istnieje pomijalna funkcja negl taka, że

$$\Pr[\text{KEav } A, \Pi(n) = 1] \leq \frac{1}{2} + \text{negl}(n).$$

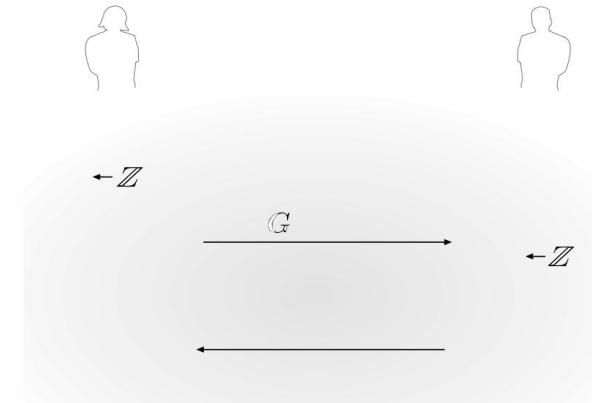
Celem protokołu wymiany kluczy jest prawie zawsze wygenerowanie wspólnego klucza k , który będzie używany przez strony do dalszych celów kryptograficznych, np. do szyfrowania i uwierzytelniania późniejszej komunikacji przy użyciu, powiedzmy, uwierzytelnionego schematu szyfrowania. Intuicyjnie użycie klucza współdzielonego wygenerowanego przez bezpieczny protokół wymiany kluczy powinno być „tak dobre, jak” użycie klucza współdzielonego za pośrednictwem kanału prywatnego. Można to udowodnić formalnie; patrz ćwiczenie 10.1.

Protokół wymiany kluczy Diffiego – Hellmana. Opiszemy teraz protokół wymiany kluczy, który pojawił się w oryginalnej pracy Diffiego i Hellmana (choćbyły one mniej formalne, niż będziemy tutaj). Niech G będzie probabilistycznym algorytmem wielomianowym w czasie, który na wejściu $1n$ wyprowadza opis grupy cyklicznej G , jej rząd q (przy $q = n$) i generator $g \in G$. (Patrz rozdział 8.3. 2.) Protokół wymiany kluczy Diffiego-Hellmana jest formalnie opisany jako Konstrukcja 10.2 i zilustrowany na rysunku 10.2.

KONSTRUKCJA 10.2

- Wspólne wejście: Parametr bezpieczeństwa $1n$
- Protokół:
 1. Alicja uruchamia $G(1n)$, aby otrzymać (G, q, g) .
 2. Alicja wybiera uniform $x \in \mathbb{Z}_q$ i oblicza $hA := g^x$. 3. Alicja wysyła (G, q, g, hA) do Boba.
 4. Bob otrzymuje (G, q, g, hA) . Wybiera mundur $y \in \mathbb{Z}_q$ i Bob wysyła hB do oblicza $hB := g^{y \cdot hA}$ Alicji i wysyła na wyjście $kB := h^{-y} \cdot hA$.
 5. Alicja odbiera hB i wyprowadza klucz $kA := h^{x-y} \cdot hB$.

Protokół wymiany kluczy Diffiego – Hellmana.



RYSUNEK 10.2: Protokół wymiany kluczy Diffiego–Hellmana.

W naszym opisie założyliśmy, że Alicja generuje (G, q, g) i wysyła te parametry do Boba w ramach swojej pierwszej wiadomości. W praktyce parametry te są wystandardyzowane, stałe i znane obu stronom przed rozpoczęciem protokołu. W takim przypadku Alicja musi jedynie wysłać hA , a Bob nie musi czekać na otrzymanie wiadomości od Alicji przed obliczeniem i wysłaniem hB .

Nietrudno zauważyc, że protokół jest poprawny: Bob oblicza klucz

$$kB = godz_A^x = (g^x)y = g^{xy}$$

a Alicja oblicza klucz

$$kA = godz_B^y = (g^y)x = sol^{xy},$$

i tak $kA = kB$. (Uważny czytelnik zauważyc, że klucz wspólnie dzielony jest elementem grupy, a nie ciągiem bitów. Powrócimy do tego punktu później.)

Diffie i Hellman nie udowodnili bezpieczeństwa swojego protokołu; rzeczywiście nie było jeszcze odpowiednich pojęć (zarówno ram definicyjnych, jak i idei formułowania precyzyjnych założeń). Zobaczmy, jakie założenia będą potrzebne, aby protokół był bezpieczny. Pierwsza obserwacja dokonana przez Diffiego i Hellmana jest taka, że minimalnym wymogiem bezpieczeństwa jest to, aby problem logarytmu dyskretnego był trudny w stosunku do G . Jeśli nie, to przeciwnik, biorąc pod uwagę transkrypcję (która w szczególności zawiera hA) może obliczyć tajną wartość jednej ze stron (tj. x), a następnie łatwo obliczyć klucz wspólnie dzielony przy użyciu tej wartości. Zatem, aby protokół był bezpieczny, konieczna jest trudność problemu logarytmu dyskretnego. Nie jest to jednak wystarczające, ponieważ możliwe jest, że istnieją inne sposoby obliczenia klucza $KA = kB$ bez jawnego obliczania x lub y . Obliczeniowe założenie Diffiego–Hellmana, które gwarantowałoby jedynie, że kluczem jest g^{xy}

trudno wyliczyć w całości na podstawie transkrypcji – to też nie wystarczy.

Definicja 10.1 wymaga, aby klucz współdzielony g xy był nie do odróżnienia od jednolitego dla dowolnego przeciwnika, biorąc pod uwagę g, g i g, dokładnie zgodnie z y. To jest decyzyjnym założeniem Diffiego-Hellmana wprowadzonym w sekcji 8.3.2.

Jak zobaczymy, dowód bezpieczeństwa protokołu wynika niemal natychmiast z decyzyjnego założenia Diffiego-Hellmana. Nie powinno to być zaskakujące, ponieważ założenia Diffiego-Hellmana wprowadzono —długo po opublikowaniu przez Diffiego i Hellmana ich artykułu —jako sposób na wyodrębnienie właściwości leżących u podstaw (domniemanego) bezpieczeństwa protokołu Diffiego-Hellmana. W związku z tym zasadne jest pytanie, czy można cokolwiek zyskać na zdefiniowaniu i udowodnieniu bezpieczeństwa w tym przypadku. Miejmy nadzieję, że w tym miejscu książki już jesteś przekonany, że odpowiedź brzmi „tak”. Precyzyjne zdefiniowanie bezpiecznej wymiany kluczy zmusza nas do przemyślenia, jakich dokładnie właściwości bezpieczeństwa potrzebujemy; określenie precyzyjnego założenia (mianowicie decyzyjnego założenia Diffiego-Hellmana) oznacza, że możemy badać to założenie niezależnie od konkretnego zastosowania i – gdy już jesteśmy przekonani o jego wiarygodności – konstruować na jego podstawie inne protokoły; wreszcie udowodnienie bezpieczeństwa pokazuje, że założenie rzeczywiście wystarczy, aby protokół spełniał nasze pożądane pojęcie bezpieczeństwa.

W naszym dowodzie bezpieczeństwa używamy zmodyfikowanej wersji definicji 10.1, w której wymagane jest, aby klucz współdzielony był nie do odróżnienia od jednolitego elementu G, a nie od jednolitego n-bitowego ciągu. Tę rozbieżność trzeba będzie wyeliminować, zanim protokół będzie mógł być zastosowany w praktyce —w końcu elementy grupowe zazwyczaj nie są przydatne jako klucze kryptograficzne, a reprezentacja jednolitego elementu grupy nie będzie, ogólnie rzecz biorąc, jednolitym ciągiem bitów —i pokrótkę omówimy jeden standardowy sposób, aby to zrobić po dowodzie. Na razie my niech KE_{A,Π(n)} oznacza zmodyfikowany eksperyment, w którym jeśli b = 1, przeciwnikowi jest dane \hat{k} wybrane równomiernie z G zamiast jednolitego n-bitowego ciągu.

TWIERDZENIE 10.3 Jeżeli decyzyjny problem Diffiego-Hellmana jest trudny w stosunku do G, to protokół wymiany kluczy Diffiego-Hellmana Π jest bezpieczny w obecności \hat{k} przeciwnika, który ^{naw} posłuchującego (w odniesieniu do zmodyfikowanego eksperymentu KE_{A,Π(n)}).

DOWÓD Niech A będzie przeciwnikiem ppt. Ponieważ $\Pr[b = 0] = \Pr[b = 1] = 1/2$, mamy

$$\begin{aligned} \Pr_{\text{KE}}^{\text{naw}} &= \Pr_{A, \Pi(n)}^{\text{naw}} = 1 \\ &= \frac{1}{2} \cdot \Pr_{\text{KE}}^{\text{naw}} \Pr_{A, \Pi(n)}^{\text{naw}} = 1 \mid b = 0 + \frac{1}{2} \cdot \Pr_{\text{KE}}^{\text{naw}} \Pr_{A, \Pi(n)}^{\text{naw}} = 1 \mid b = 1. \end{aligned}$$

W eksperymencie KE_{A,Π(n)} przeciwnik A otrzymuje (G, q, g, hA, hB, \hat{k}), gdzie (G, q, g, hA, hB) reprezentuje transkrypcję wykonania protokołu, a \hat{k} jest albo faktycznym kluczem wyliczonym przez strony (jeśli b = 0) lub jednolitym elementem grupowym (jeśli b = 1). Rozróżnienie tych dwóch przypadków jest dokładne

równoważne rozwiązaniu decyzyjnego problemu Diffiego – Hellmana. To jest

$$\begin{aligned}
 & \Pr_{\text{KE}}^{\text{naw}} A, \Pi(n) = 1 \\
 &= \frac{1}{2} \cdot \Pr_{\text{KE}}^{\text{naw}} A, \Pi(n) = 1 \mid b = 0 + \frac{1}{2} \cdot \Pr_{\text{KE}}^{\text{naw}} A, \Pi(n) = 1 \mid b = 1 \\
 &= \frac{1}{2} \left[\Pr[A(G, g, q, gx, gy, gxy) = 0] + \Pr[A(G, q, g, gx, gy, gz) = 1] \right] \\
 &= \frac{1}{2} \left[\Pr[A(G, g, q, gx, gy, gxy) = 1] + \Pr[A(G, q, g, gx, gy, gz) = 1] \right] \\
 &= \frac{1}{2} \left[\Pr[A(G, g, q, gx, gy, gz) = 1] - \Pr[A(G, q, g, gx, gy, gxy) = 1] \right] \\
 &\quad + \Pr[A(G, g, q, gx, gy, gz) = 1] - \Pr[A(G, q, g, gx, gy, gxy) = 1],
 \end{aligned}$$

gdzie wszystkie prawdopodobieństwa są przejmowane (G, q, g) i wyprowadzane przez $G(1n)$ oraz równomierny wybór $x, y, z \in \mathbb{Z}_q$. (Zauważ, że ponieważ g jest mundur jest generatorem, g elementem G , gdy z jest równomiernie rozłożone w \mathbb{Z}_q .) Jeśli decyzyjne założenie Diffiego-Hellmana jest trudne w stosunku do G , oznacza to dokładnie, że istnieje pomijalna funkcja negl, dla której

$$\Pr[A(G, g, q, gx, gy, gz) = 1] - \Pr[A(G, q, g, gx, gy, gxy) = 1] = \text{negl}(n).$$

Dochodzimy do wniosku, że

$$\Pr_{\text{KE}}^{\text{naw}} A, \Pi(n) = 1 = \frac{1}{2} + \frac{1}{2} \cdot \text{zaniebywać}(n),$$

uzupełnienie dowodu. ■

Jednolite elementy grupowe a jednolite ciągi bitowe. Poprzednie twierdzenie pokazuje, że kluczowy wynik Alicji i Boba w protokole Diffiego-Hellmana jest nie do odróżnienia (dla podsłuchiwacza w czasie wielomianowym) od jednolitego elementu grupy. Aby móc używać klucza w kolejnych zastosowaniach kryptograficznych —a także spełniać definicję 10.1 —wynik klucza uzyskany przez strony powinien być nie do odróżnienia od jednolitego ciągu bitów o odpowiedniej długości. Protokół Diffiego-Hellmana można zmodyfikować, aby to osiągnąć, poprzez zastosowanie przez strony odpowiedniej funkcji wyprowadzania klucza (por. sekcja 5.6.4) do obliczanego przez siebie elementu grupy współdzielonej g^{xy} .

Aktywni przeciwnicy. Do tej pory rozważaliśmy jedynie przeciwnika podsłuchującego. Chociaż ataki podsłuchowe są zdecydowanie najczęstsze (ponieważ są najłatwiejsze do przeprowadzenia), w żadnym wypadku nie są jedynym możliwym atakiem. Aktywne ataki, podczas których przeciwnik wysyła własne wiadomości do jednej lub obu stron, również stanowią problem, a każdy protokół stosowany w praktyce musi być również odporny na takie ataki. Rozważając ataki aktywne, warto nieformalnie rozróżnić ataki polegające na podszywaniu się pod inne osoby

przeciwnik podszywa się pod jedną ze stron podczas interakcji z drugą stroną oraz ataki typu man-in-the-middle, podczas których obie uczciwe strony wykonują protokół, a przeciwnik przechwytuje i modyfikuje wiadomości wysyłane od jednej strony do drugiej. Nie będziemy formalnie definiować bezpieczeństwa przed żadną z klas ataków, ponieważ takie definicje są dość skomplikowane i nie można ich osiągnąć bez wcześniejszego udostępnienia przez strony pewnych informacji. Niemniej jednak warto zauważyć, że protokół Diffiego-Hellmana jest całkowicie niezabezpieczony przed atakami typu man-in-the-middle. W rzeczywistości przeciwnik typu „man-in-the-middle” może działać w taki sposób, że Alicja i Bob kończą protokół różnymi kluczami kA i kB , które są znane przeciwnikowi, ale ani Alicja, ani Bob nie są w stanie wykryć, że jakkolwiek przeprowadzono atak. Szczegóły tego ataku pozostawiamy jako ćwiczenie.

Wymiana kluczy Diffiego-Hellmana w praktyce. Protokół Diffiego-Hellmana w swojej podstawowej formie zwykle nie jest stosowany w praktyce ze względu na brak bezpieczeństwa przed atakami typu man-in-the-middle, jak omówiono powyżej. Nie umniejsza to w żaden sposób jego wagi. Protokół Diffiego-Hellmana posłużył jako pierwsza demonstracja, że techniki asymetryczne (i problemy z teorią liczb) można wykorzystać do złagodzenia problemów z dystrybucją kluczy w kryptografii.

Co więcej, protokół Diffiego-Hellmana stanowi rdzeń standardowych protokołów wymiany kluczy, które są odporne na ataki typu man-in-the-middle i są obecnie szeroko stosowane. Jednym z godnych uwagi przykładów jest TLS; patrz sekcja 12.8.

10.4 Rewolucja klucza publicznego

Oprócz wymiany kluczy Diffie i Hellman wprowadzili w swojej przełomowej pracy także pojęcie kryptografii z kluczem publicznym (czyli asymetrycznej).

W ustawieniu klucza publicznego (w przeciwnieństwie do ustawienia klucza prywatnego, które badaliśmy w rozdziałach 1–7) strona chcącą bezpiecznie się komunikować generuje parę kluczy: klucz publiczny, który jest szeroko rozpowszechniony, i klucz prywatny, że trzyma to w tajemnicy. (Fakt, że obecnie istnieją dwa różne klucze, powoduje, że schemat jest asymetryczny). Po wygenerowaniu tych kluczy strona może ich użyć, aby zapewnić poufność otrzymywanych wiadomości przy użyciu schematu szyfrowania kluczem publicznym lub integralność wiadomości wysyłanych przy użyciu schematu schemat podpisu cyfrowego. (Patrz rysunek 10.3.) Przedstawiamy tutaj krótki opis tych prymitywów i szczegółowo omawiamy je odpowiednio w rozdziałach 11 i 12.

W schemacie szyfrowania klucza publicznego klucz publiczny wygenerowany przez jakąś stronę służy jako klucz szyfrowania; każdy, kto zna ten klucz publiczny, może go użyć do szyfrowania wiadomości i generowania odpowiednich szyfrogramów. Klucz prywatny służy jako klucz deszyfrujący i jest używany przezznającą go stronę do odzyskania oryginalnej wiadomości z dowolnego tekstu zaszyfrowanego wygenerowanego przy użyciu pasującego klucza publicznego. Co więcej — i to zdumiewające, że coś takiego istnieje! —

	Ustawienie klucza prywatnego	Ustawienie klucza publicznego
Tajność	Szyfrowanie kluczem prywatnym	Szyfrowanie kluczem publicznym
Integralność	Kody uwierzytelniania wiadomości	Schematy podpisu cyfrowego

RYSUNEK 10.3: Podstawowe elementy kryptograficzne w ustawieniach klucza prywatnego i publicznego.

tajemnica zaszyfrowanych wiadomości jest zachowana nawet przed przeciwnikiem, który zna klucz szyfrowania (ale nie klucz deszyfrujący). Innymi słowy, (publiczny) klucz szyfrujący jest bezużyteczny dla atakującego próbującego odszyfrować teksty zaszyfrowane przy użyciu tego klucza. Aby umożliwić tajną komunikację, odbiorca może po prostu wysłać swój klucz publiczny potencjalnemu nadawcy (bez martwienia się o podsłuchującego przeciwnika, który zauważa jej klucz publiczny) lub opublikować swój klucz publiczny na swojej stronie internetowej lub w jakiejś centralnej bazie danych. Schemat szyfrowania kluczem publicznym umożliwia zatem prywatną komunikację bez polegania na prywatnym kanale dystrybucji kluczy.¹ Schemat podpisu cyfrowego jest analogiem klucza publicznego do kodów uwierzytelniania wiadomości (MAC). W tym przypadku klucz prywatny służy jako „klucz uwierzytelniający”

(częściej nazywany kluczem podpisującym), który umożliwia stronie znającej ten klucz generowanie „znaczników uwierzytelniających” (tj. podpisów) dla wysyłanych wiadomości.

Klucz publiczny pełni funkcję klucza weryfikacyjnego, umożliwiając każdemu, kto go zna, weryfikację podpisów wydanych przez nadawcę. Podobnie jak w przypadku komputerów MAC, można zastosować schemat podpisu cyfrowego, aby zapobiec niewykrytemu manipulowaniu wiadomością. Fakt, że weryfikacji może dokonać każdy, kto zna klucz publiczny nadawcy, okazuje się jednak mieć daleko idące konsekwencje. W szczególności umożliwia pobranie dokumentu podpisanego przez Alicję i przedstawienie go stronie trzeciej (powiedzmy sądu) jako dowód, że Alicja rzeczywiście podpisała dokument. Ta właściwość nazywa się niezaprzecalnością i ma szerokie zastosowanie w handlu elektronicznym. Możliwe jest na przykład cyfrowe podpisywanie umów, wysyłanie podpisanych elektronicznych zamówień zakupu lub obietnic płatności i tak dalej.

Podpisy cyfrowe są również wykorzystywane do bezpiecznej dystrybucji kluczy publicznych w ramach infrastruktury klucza publicznego, co omówiono bardziej szczegółowo w sekcji 12.7.

W swoim artykule Diffie i Hellman przedstawili pojęcie kriptografii klucza publicznego, ale nie przedstawili żadnych potencjalnych konstrukcji. Rok później Ron Rivest, Adi Shamir i Len Adleman zaproponowali problem RSA i przedstawili pierwszą publiczną -kluczowe schematy szyfrowania i podpisu cyfrowego w oparciu o stopień trudności tego problemu. Warianty ich schematów należą obecnie do najpowszechniej używanych prymitywów kryptograficznych. W 1985 roku Taher El Gamal przedstawił schemat szyfrowania, który w zasadzie stanowi niewielką odmianę protokołu wymiany kluczy Diffie-Hellman i jest obecnie również szeroko stosowany. Zatem chociaż

¹ Na razie jednak zakładamy, że jest to kanał uwierzytelniony, który pozwala nadawcy uzyskać legalną kopię klucza publicznego odbiorcy. W sekcji 12.7 pokazujemy, jak można zastosować kriptografię klucza publicznego, aby zmniejszyć również to założenie.

Diffiemu i Hellmanowi nie udało się skonstruować (nieinteraktywnego) schematu szyfrowania klucza publicznego, byli bardzo blisko.

Zakończymy podsumowaniem, w jaki sposób kryptografia klucza publicznego radzi sobie z ograniczeniami ustawię klucza prywatnego omówionymi w sekcji 10.1:

1. Szyfrowanie kluczem publicznym umożliwia dystrybucję kluczy za pośrednictwem kanałów publicznych (ale uwierzytelnych). Może to uprościć dystrybucję i aktualizację kluczowych materiałów.
2. Kryptografia klucza publicznego zmniejsza potrzebę przechowywania przez użytkowników wielu tajnych kluczy. Rozważmy ponownie sytuację w dużej korporacji, w której każda para pracowników potrzebuje możliwości bezpiecznej komunikacji. Stosując kryptografię klucza publicznego, wystarczy, aby każdy pracownik przechowywał tylko jeden klucz prywatny (własny) i klucze publiczne wszystkich pozostałych pracowników. Co ważne, te ostatnie klucze nie muszą być przechowywane w tajemnicy; można je nawet przechowywać w jakimś centralnym (publicznym) repozytorium.
3. Wreszcie, kryptografia klucza publicznego jest (bardziej) odpowiednia dla środowisk otwartych, w których strony, które nigdy wcześniej nie miały ze sobą interakcji, chcą mieć możliwość bezpiecznej komunikacji. Jednym z typowych przykładów jest to, że sprzedawca internetowy może opublikować swój klucz publiczny w Internecie; użytkownik dokonujący zakupu może następnie uzyskać klucz publiczny sprzedawcy, jeśli zajdzie taka potrzeba, gdy będzie musiał zaszyfrować dane swojej karty kredytowej.

Wynalezienie szyfrowania kluczem publicznym było rewolucją w kryptografii.

To nie przypadek, że do późnych lat 70. i wczesnych 80. szyfrowanie i kryptografia w ogóle należały do domeny wywiadu i organizacji wojskowych i dopiero wraz z pojawiением się technik klucza publicznego zastosowanie kryptografii rozprzestrzeniło się na masy.

Referencje i dodatkowe lektury

Problemy dystrybucji kluczy i zarządzania kluczami omówiliśmy jedynie pokrótko. Aby uzyskać więcej informacji, zalecamy zapoznanie się z podręcznikami dotyczącymi bezpieczeństwa sieci. Kaufmana i in. [102] dobrze opisują protokoły bezpiecznej dystrybucji kluczy, ich cel i sposób działania.

Nie podejmowaliśmy żadnych prób uchwycenia pełnej historii rozwoju kryptografii klucza publicznego. Inni, poza Diffie i Hellmanem, pracowali nad podobnymi pomysłami w latach 70. Szczególnie jednym z badaczy wykonujących podobną i niezależną pracę był Ralph Merkle, przez wielu uważany za współtwórcę kryptografii klucza publicznego (choćże publikował po Diffie i Hellmanie). Wspominamy także Michaela Rabina, który około roku po pracach Rivesta, Shamira i Adlemana opracował konstrukcje schematów podpisów i schematów szyfrowania klucza publicznego w oparciu o trudność faktoringu [148].

Gorąco polecamy przeczytanie oryginalnej pracy Diffiego i Hellmana [58] oraz odesłanie czytelnika do książki Levy'ego [114], aby uzyskać więcej informacji na temat politycznych i historycznych aspektów rewolucji klucza publicznego.

Co ciekawe, aspekty kryptografii klucza publicznego odkryto w społeczności wywiadowczej, zanim opublikowano je w otwartej literaturze naukowej.

We wczesnych latach siedemdziesiątych James Ellis, Clifford Cocks i Malcolm Williamson z brytyjskiej agencji wywiadowczej GCHQ wymyślili pojęcie kryptografii klucza publicznego, wariant szyfrowania RSA i wariant protokołu wymiany kluczy Diffiego – Hellmana. Ich prace odtajniono dopiero w 1997 r. Choć podstawy matematyki leżące u podstaw kryptografii klucza publicznego mogły zostać odkryte przed 1976 r., można śmiało powiedzieć, że szerokie konsekwencje tej nowej technologii doceniono dopiero, gdy pojawiły się Diffie i Hellman.

Ćwiczenia

10.1 Niech Π będzie protokołem wymiany kluczy, a (Enc, Dec) będzie schematem szyfrowania klucza prywatnego. Rozważmy następujący protokół interaktywny Π do szyfrowania wiadomości: najpierw nadawca i odbiorca uruchamiają Π , aby wygenerować wspólny klucz k . Następnie nadawca oblicza $c = \text{Enck}(m)$ i wysyła c drugiej stronie, która odszyfrowuje i otrzymując m za pomocą k .

(a) Sformułuj definicję szyfrowania nieroźróżnialnego w obecności osoby podsłuchującej (por. Definicja 3.8) odpowiednią dla tego interaktywnego ustalenia. (b)

Udowodnij, że jeśli Π jest bezpieczny w obecności podsłuchującego i (Enc, Dec) ma nieroźróżnialne szyfrowanie w obecności podsłuchującego, to Π spełnia Twoją definicję.

10.2 Pokaż, że dla każdej z grup rozważanych w rozdziałach 8.3.3 lub 8.3.4 jednolity element grupy (wyrażony za pomocą reprezentacji naturalnej) jest łatwo odróżnialny od jednolitego ciągu bitów o tej samej długości.

10.3 Opisz atak typu man-in-the-middle na protokół Diffiego-Hellmana, w którym przeciwnik dzieli klucz KA z Alicją i (inny) klucz KB z Bobem, a Alicja i Bob nie mogą wykryć, że coś jest nie tak.

10.4 Rozważ następujący protokół wymiany kluczy:

(a) Alicja wybiera jednostajne $k, r \in \{0, 1\}^n$ i wysyła $s := k \oplus r$ do Boba. (b) Bob wybiera uniform $t \in \{0, 1\}^n$ i wysyła $u := s \oplus t$ do Alicji. (c) Alicja oblicza $w := u \oplus r$ i wysyła w do Boba. (d) Alicja wyprowadza k , a Bob wyprowadza $w \oplus t$.

Pokaż, że Alicja i Bob wyprowadzają ten sam klucz. Przeanalizuj bezpieczeństwo schematu (tj. udowodnij jego bezpieczeństwo lub pokaż konkretny atak).

Machine Translated by Google

Rozdział 11

Szyfrowanie kluczem publicznym

11.1 Szyfrowanie kluczem publicznym – przegląd

Wprowadzenie szyfrowania kluczem publicznym oznaczało rewolucję w kryptografii. Do tego czasu kryptografowie polegali wyłącznie na współdzielonych, tajnych kluczach, aby osiągnąć prywatną komunikację. Z kolei techniki klucza publicznego umożliwiają stronom prywatną komunikację bez wcześniejszego uzgadniania jakichkolwiek tajnych informacji. Jak już zauważaliśmy, jest to dość zdumiewające i sprzeczne z intuicją, że jest to możliwe: oznacza to, że dwie osoby po przeciwnych stronach pokoju, które mogą porozumiewać się jedynie poprzez krzyczenie do siebie i nie mają początkowej tajemnicy, mogą rozmawiać w taki sposób aby nikt inny na sali nie dowiedział się niczego o tym, co mówią!

Podczas ustawiania szyfrowania kluczem prywatnym dwie strony ustalają tajny klucz, którego każda ze stron może używać zarówno do szyfrowania, jak i deszyfrowania. Szyfrowanie kluczem publicznym jest asymetryczne w obu tych aspektach. Jedna strona (odbiorca) generuje parę kluczy (pk , sk), zwanych odpowiednio kluczem publicznym i kluczem prywatnym. Klucz publiczny jest używany przez nadawcę do szyfrowania wiadomości; odbiorca używa klucza prywatnego do odszyfrowania powstałego tekstu zaszyfrowanego.

Ponieważ celem jest uniknięcie konieczności wcześniejszego spotkania się dwóch stron w celu uzgodnienia jakichkolwiek informacji, w jaki sposób nadawca uczy się pk ? Na poziomie abstrakcyjnym może to nastąpić na dwa sposoby. Zadzwój do odbiorcy Alice i nadawcy Bob. W pierwszym podejściu, gdy Alicja dowiaduje się, że Bob chce się z nią porozumieć, może w tym momencie wygenerować (pk , sk) (zakładając, że jeszcze tego nie zrobiła), a następnie wysłać pk do Boba w sposób jawnym; Bob może następnie użyć pk do zaszyfrowania swojej wiadomości. Podkreślamy, że kanał pomiędzy Alicją a Bobem może być publiczny, ale zakłada się, że jest uwierzytelniony, co oznacza, że przeciwnik nie może modyfikować klucza publicznego wysłanego przez Alicję do Boba (a w szczególności nie może go zastąpić własnym kluczem). W sekcji 12.7 omówiono, w jaki sposób klucze publiczne mogą być dystrybuowane w nieuwierzytelnionych kanałach.

Alternatywnym podejściem jest wcześniejsze wygenerowanie przez Alicję kluczy (pk , sk), niezależnie od konkretnego nadawcy. (W rzeczywistości w czasie generowania klucza Alicja nie musi nawet być świadoma, że Bob chce z nią rozmawiać ani nawet, że Bob istnieje.) Alicja może szeroko rozpowszechnić swój klucz publiczny, powiedzmy, publikując go na swojej stronie internetowej, umieszczając umieścić go na swoich wizytówkach lub umieścić w publicznym spisie ludności. Teraz każdy, kto chce prywatnie komunikować się z Alicją, może to zrobić

sprawdź jej klucz publiczny i postępuj jak powyżej. Należy pamiętać, że wielu nadawców może wielokrotnie komunikować się z Alicją, używając tego samego klucza publicznego pk do szyfrowania całej swojej komunikacji.

Należy pamiętać, że pk jest z natury publiczne —a zatem atakujący może go łatwo poznać — w każdym z powyższych scenariuszy. W pierwszym przypadku przeciwnik podsłuchujący komunikację Alicji i Boba uzyskuje bezpośrednio pk; w drugim przypadku przeciwnik również dobrze mógłby samodzielnie sprawdzić klucz publiczny Alicji. Widzimy, że bezpieczeństwo szyfrowania klucza publicznego nie może opierać się na tajemnicy pk, ale zamiast tego musi opierać się na tajemnicy sk. Dlatego ważne jest, aby Alicja nie ujawniała swojego klucza prywatnego nikomu, łącznie z nadawcą Bobem.

Porównanie z szyfrowaniem kluczem prywatnym

Być może najbardziej oczywistą różnicą między szyfrowaniem kluczem prywatnym i publicznym jest to, że w pierwszym przypadku zakładana jest całkowita tajność wszystkich kluczy kryptograficznych, podczas gdy w drugim wymagane jest zachowanie tajemnicy tylko w przypadku klucza prywatnego sk. Choć może się to wydawać niewielką różnicą, konsekwencje są ogromne: w przypadku ustawienia klucza prywatnego komunikujące się strony muszą w jakiś sposób być w stanie udostępnić tajny klucz, nie pozwalając osobom trzecim na poznanie go, podczas gdy w przypadku ustawienia klucza publicznego klucz publiczny można przesyłać od jednej strony do drugiej kanałem publicznym bez narażenia bezpieczeństwa. W przypadku osób krzyczących przez pokój lub, bardziej realistycznie, komunikujących się za pośrednictwem sieci publicznej, takiej jak linia telefoniczna lub Internet, jedyną opcją jest szyfrowanie kluczem publicznym.

Inną ważną różnicą jest to, że schematy szyfrowania z kluczem prywatnym używają tego samego klucza zarówno do szyfrowania, jak i deszyfrowania, podczas gdy schematy szyfrowania z kluczem publicznym używają różnych kluczy do każdej operacji. Oznacza to, że szyfrowanie klucza publicznego jest z natury asymetryczne. Ta asymetria w ustawieniu klucza publicznego oznacza, że role nadawcy i odbiorcy nie są wymienne, tak jak ma to miejsce w przypadku ustawienia klucza prywatnego: pojedyncza para kluczy umożliwia komunikację tylko w jednym kierunku. (Komunikację dwukierunkową można osiągnąć na wiele sposobów; chodzi o to, że pojedyncze wywołanie schematu szyfrowania klucza publicznego wymusza rozróżnienie między jednym użytkownikiem pełniącym rolę odbiorcy a innymi użytkownikami pełniącymi rolę nadawcy). pojedynczy przypadek schematu szyfrowania kluczem publicznym umożliwia wielu nadawcom komunikację prywatną z jednym odbiorcą, w przeciwieństwie do przypadku klucza prywatnego, w którym tajny klucz współdzielony między dwiema stronami umożliwia prywatną komunikację tylko między tymi dwiema stronami.

Podsumowując i opracowując poprzednią dyskusję, widzimy, że szyfrowanie kluczem publicznym ma następujące zalety w porównaniu z szyfrowaniem kluczem prywatnym:

- Szyfrowanie kluczem publicznym rozwiązuje (w pewnym stopniu) problem dystrybucji kluczy, ponieważ komunikujące się strony nie muszą w tajemnicy dzielić się kluczem przed komunikacją. Dwie strony mogą porozumiewać się w tajemnicy, nawet jeśli cała komunikacja między nimi jest monitorowana.
- Kiedy pojedynczy odbiorca komunikuje się z N nadawcami (np. sprzedawca internetowy przetwarzający zamówienia kart kredytowych od wielu kupujących), jest to

znacznie wygodniejsze dla odbiorcy jest przechowywanie pojedynczego klucza prywatnego sk zamiast udostępniania, przechowywania i zarządzania N różnymi tajnymi kluczami (tj. po jednym dla każdego nadawcy). W rzeczywistości w przypadku szyfrowania kluczem publicznym liczba i tożsamość potencjalnych nadawców nie muszą być znane w momencie generowania klucza. Pozwala to na ogólną elastyczność w „systemach otwartych”.

Fakt, że schematy szyfrowania z kluczem publicznym pozwalają każdemu działać w roli nadawcy, może być wadą, gdy odbiorca chce otrzymywać wiadomości tylko od jednej konkretnej osoby. W takim przypadku schemat szyfrowania uwierzytelnionego (klucza prywatnego) byłby lepszym wyborem niż szyfrowanie kluczem publicznym.

Główna wadą szyfrowania kluczem publicznym jest to, że jest ono około 2 do 3 rządów wielkości wolniejsze niż szyfrowanie kluczem prywatnym.¹ Zaimplementowanie szyfrowania kluczem publicznym w urządzeniach o znacznie ograniczonych zasobach, takich jak karty inteligentne lub urządzenia wykorzystujące częstotliwość radiową, może stanowić wyzwanie. znaczniki identyfikacyjne (RFID). Nawet jeśli komputer stacjonarny wykonuje operacje kryptograficzne, wykonywanie tysięcy takich operacji na sekundę (jak w przypadku sprzedawcy internetowego przetwarzającego transakcje kartą kredytową) może być zaporowe. Zatem, jeśli istnieje możliwość szyfrowania klucza prywatnego (tj. jeśli dwie strony mogą wcześniej bezpiecznie udostępnić klucz), wówczas zazwyczaj należy z niego skorzystać.

W rzeczywistości, jak zobaczymy w sekcji 11.3, szyfrowanie kluczem prywatnym jest używane w ustawieniu klucza publicznego w celu poprawy wydajności szyfrowania (kluczem publicznym) długich wiadomości. Dogłębne zrozumienie szyfrowania kluczem prywatnym jest zatem niezbędne, aby docenić sposób, w jaki szyfrowanie kluczem publicznym jest wdrażane w praktyce.

Bezpieczna dystrybucja kluczy publicznych

W całej naszej dotychczasowej dyskusji domyślnie założyliśmy, że przeciwnik jest pasywny; to znaczy, że przeciwnik jedynie podsłuchuje komunikację między nadawcą i odbiorcą, ale nie zakłóca aktywnie komunikacji. Jeśli przeciwnik ma możliwość manipulowania całą komunikacją pomiędzy uczciwymi stronami, a te uczciwe strony nie dzielą się wcześniej kluczami, wówczas prywatności po prostu nie da się osiągnąć. Na przykład, jeśli odbiorca Alicja wyśle Bobowi swój klucz publiczny pk, ale przeciwnik zastąpi go własnym kluczem pk (dla którego zna pasujący klucz prywatny sk), to nawet jeśli Bob zaszyfruje swoją wiadomość za pomocą pk, przeciwnik to zrobi łatwo odzyskać wiadomość (używając sk). Podobny atak działa, jeśli przeciwnik jest w stanie zmienić wartość klucza publicznego Alicji przechowywanego w jakimś katalogu publicznym lub jeśli przeciwnik może manipulować kluczem publicznym przesyłanym z katalogu publicznego do Boba. Jeśli Alicja i Bob nie podzielą się z wyprzedzeniem żadnymi informacjami i nie chcą polegać na jakiejś wzajemnie zaufanej stronie trzeciej, Alicja i Bob nie mogą nic zrobić, aby zapobiec aktywnym atakom tego rodzaju, lub

¹Trudno jest podać dokładne porównanie, ponieważ względna wydajność zależy od konkretnych rozważanych schematów, a także różnych szczegółów wdrożenia.

nawet powiedzieć, że taki atak ma miejsce.²

Co ważne, w naszym podejściu do szyfrowania klucza publicznego w tym rozdziale zakładamy, że nadawcy są w stanie uzyskać legalną kopię klucza publicznego odbiorcy.

(Będzie to ukryte w dostarczonych przez nas definicjach zabezpieczeń.) Oznacza to, że zakładamy bezpieczną dystrybucję kluczy. Założenie to przyjęto nie dlatego, że aktywne ataki omawianego powyżej typu nie budzą obaw —w rzeczywistości stanowią one poważne zagrożenie, z którym należy się uporać w każdym rzeczywistym systemie korzystającym z szyfrowania kluczem publicznym. Założenie to przyjęto raczej dlatego, że istnieją inne mechanizmy zapobiegania aktywnym atakom (patrz na przykład sekcja 12.7) i dlatego wygodne (i użyteczne) jest oddzielenie badań nad bezpiecznym szyfrowaniem klucza publicznego od badania nad bezpieczna dystrybucja klucza publicznego.

11.2 Definicje

Zaczynamy od zdefiniowania składni szyfrowania kluczem publicznym. Definicja jest bardzo podobna do definicji 3.7, z tą różnicą, że zamiast pracować tylko z jednym kluczem, mamy teraz odrębne klucze szyfrujące i deszyfrujące.

DEFINICJA 11.1 Schemat szyfrowania klucza publicznego to potrójny probabilistyczny algorytm wielomianowy (Gen , Enc , Dec), taki że:

1. Algorytm generowania klucza Gen przyjmuje na wejściu parametr bezpieczeństwa n i ¹wyprowadza parę kluczy (pk, sk) . Pierwszy z nich nazywamy kluczem publicznym, a drugi kluczem prywatnym. Zakładamy dla wygody, że każdy pk i sk ma długość co najmniej n i że n można wyznaczyć z pk , sk .
2. Algorytm szyfrowania Enc przyjmuje jako dane wejściowe klucz publiczny pk i wiadomość m z pewnej przestrzeni wiadomości (która może zależeć od pk). Wysyła zaszyfrowany tekst c i zapisujemy go jako $c = \text{Enc}_{\text{pk}}(m)$. (Patrząc w przyszłość, Enc będzie musiało działać probabilistycznie, aby osiągnąć znaczące bezpieczeństwo.)
3. Deterministyczny algorytm deszyfrujący Dec przyjmuje jako dane wejściowe klucz prywatny sk i tekst zaszyfrowany c , a na wyjściu wysyła wiadomość m lub specjalny symbol \perp oznaczający niepowodzenie. Zapisujemy to jako $m := \text{Dec}_{\text{sk}}(c)$.

Wymagane jest, aby, z wyjątkiem ewentualnie znikomego prawdopodobieństwa wystąpienia (pk, sk) wyniku przez $\text{Gen}(1n)$, mieliśmy $\text{Dec}_{\text{sk}}(\text{Enc}_{\text{pk}}(m)) = m$ dla dowolnego (legalnego) komunikatu m .

Wałą różnicą w stosunku do ustawnienia klucza prywatnego jest to, że algorytm generowania klucza Gen generuje teraz dwa klucze zamiast jednego. Publiczność

²W naszym scenariuszu „krzyzienia przez cały pokój” Alicja i Bob potrafią wykryć, kiedy przeciwnik zakłóca komunikację. Dzieje się tak tylko dlatego, że: (1) przeciwnik nie może uniemożliwić wiadomości Alicji dotarcia do Boba oraz (2) Alicja i Bob „udostępniają się” z wyprzedzeniem informacjami (np. dźwiękiem ich głosów), które pozwalają im „uwierzytelnić” ich komunikację.

klucz pk służy do szyfrowania, natomiast klucz prywatny sk służy do deszyfrowania.

Powtarzając naszą wcześniejszą dyskusję, zakłada się, że pk jest szeroko rozpowszechnione, więc każdy może szyfrować wiadomości dla strony, która wygenerowała ten klucz, ale sk musi być prywatne przez odbiorcę, aby ewentualnie zachować bezpieczeństwo.

Dopuszczamy znikome prawdopodobieństwo błędu deszyfrowania i rzeczywiście niektóre z przedstawionych przez nas schematów będą miały znikome prawdopodobieństwo błędu (np. jeśli trzeba wybrać liczbę pierwszą, ale z znakomitym prawdopodobieństwem zamiast tego otrzymany zostanie złożony). Mimo to odtąd generalnie będziemy ignorować tę kwestię.

W celu praktycznego wykorzystania szyfrowania kluczem publicznym będziemy chcieli, aby przestrzeń wiadomości wynosiła $\{0, 1\}$ n lub $\{0, 1\}^n$ (a w szczególności była niezależna od klucza publicznego). Chociaż czasami będziemy opisywać schematy szyfrowania wykorzystujące pewną przestrzeń komunikatów M, która nie zawiera wszystkich ciągów bitów o określonej długości (i to może również zależeć od klucza publicznego), w takich przypadkach określmy również sposób kodowania ciągów bitów jako elementy M. To kodowanie musi być zarówno wydajne obliczeniowo, jak i skutecznie odwracalne, aby odbiornik mógł odzyskać zaszyfrowany串 bitów.

11.2.1 Bezpieczeństwo przed atakami w postaci wybranego tekstu jawnego

Nasze podejście do bezpieczeństwa rozpoczęmy od wprowadzenia „naturalnego” odpowiednika Definicji 3.8 w ustawieniu klucza publicznego. Ponieważ obszerne uzasadnienie tej definicji (jak również innych, które zobaczymy) zostało już podane w Rozdziale 3, dyskusja w tym miejscu będzie stosunkowo krótka i skupi się przede wszystkim na różnicach pomiędzy ustawieniami klucza prywatnego i klucza publicznego.

Biorąc pod uwagę schemat szyfrowania klucza publicznego $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ i przeciwnika A, rozważ następujący eksperyment:

Eksperyment z nierozróżnialnością podsłuchu PubKeav $A, \Pi(n)$:

1. Uruchomiono $\text{Gen}(1n)$ w celu uzyskania kluczy (pk, sk).
2. Przeciwnik A otrzymuje pk i wysyła parę komunikatów o równej długości m_0, m_1 w przestrzeni komunikatów.
3. Wybierany jest jednolity bit $b \in \{0, 1\}$, a następnie obliczany jest zaszyfrowany tekst c = $\text{Enc}_{\text{pk}}(mb)$ i przekazywany A. Nazywamy c szyfrogramem prowokacyjnym.
4. A wyprowadza trochę b. Wynik eksperymentu wynosi 1, jeśli $b = b$, i 0 w przeciwnym razie. Jeśli $b = b$, mówimy, że A się udało.

DEFINICJA 11.2 Schemat szyfrowania kluczem publicznym $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ charakteryzuje się nierozróżnialnym szyfrowaniem w obecności podsłuchującego, jeśli dla wszystkich probabilistycznych przeciwników A w czasie wielomianowym istnieje pomijalna funkcja negl taka, że

$$\Pr[\text{PubKeav } A, \Pi(n) = 1] = \frac{1}{2} + \text{negl}(n).$$

Główna różnica pomiędzy powyższą definicją a definicją 3.8 polega na tym, że tutaj A otrzymuje klucz publiczny pk. Co więcej, pozwalamy A wybrać swoje komunikaty m0 i m1 w oparciu o ten klucz publiczny. Jest to istotne przy definiowaniu bezpieczeństwa szyfrowania klucza publicznego, ponieważ, jak omówiono wcześniej, zakładamy, że przeciwnik zna klucz publiczny odbiorcy.

Pozornie „niewielka” modyfikacja polegająca na przekazaniu przeciwnikowi A klucza publicznego pk używanego do szyfrowania wiadomości ma ogromny wpływ: w rzeczywistości daje A dostęp do wyroczni szyfrującej za darmo. (Koncepcja wyroczni szyfrującej została wyjaśniona w Sekcji 3.4.2.) Jest to prawdą, ponieważ przeciwnik, mając dane pk, może samodzielnie zaszyfrować dowolną wiadomość m, po prostu obliczając $\text{Encpk}(m)$. (Jak zawsze zakłada się, że A zna algorytm Enc.)

Wynik jest taki, że definicja 11.2 jest równoważna bezpieczeństwu CPA (tj. zabezpieczeniu przed atakami z użyciem wybranego tekstu jawnego), zdefiniowanemu w sposób analogiczny do definicji 3.22, z jedyną różnicą jest to, że atakujący otrzymuje klucz publiczny w odpowiednim eksperymentie. Mamy zatem:

PROPOZYCJA 11.3 Jeżeli schemat szyfrowania z kluczem publicznym ma szyfrowanie nieroróżnialne w obecności osoby podsłuchującej, jest on bezpieczny zgodnie z CPA.

Kontrastuje to z ustawieniem klucza prywatnego, gdzie istnieją schematy, które mają nieroróżnialne szyfrowanie w obecności podsłuchującego, ale są niepewne w przypadku ataku wybranego tekstu jawnego (patrz Propozycja 3.20). Dalsze różnice w stosunku do ustawienia klucza prywatnego, które pojawiają się niemal natychmiast jako konsekwencje powyższego, zostaną omówione dalej.

Niemożność doskonale tajnego szyfrowania klucza publicznego. Całkowicie tajne szyfrowanie klucza publicznego można zdefiniować analogicznie do definicji 2.3, uzależniając od całego widoku osoby podsłuchującej (tj. łącznie z kluczem publicznym). Również można to zdefiniować, rozszerzając Definicję 11.2 tak, aby wymagała, aby dla wszystkich przeciwników A (nie tylko skutecznych) mamy:

$$\Pr[\text{PubKeav } A, \Pi(n) = 1] = \frac{1}{2}.$$

Jednak w przeciwnieństwie do ustawienia klucza prywatnego, całkowicie tajne szyfrowanie kluczem publicznym jest niemożliwe, niezależnie od tego, jak długie są klucze i jak mała jest przestrzeń wiadomości. W rzeczywistości nieograniczony przeciwnik mający podane pk i zaszyfrowany tekst c obliczony poprzez $c = \text{Encpk}(m)$ może wyznaczyć m z prawdopodobieństwem 1. Dowód pozostaje to jako ćwiczenie 11.1.

Niepewność deterministycznego szyfrowania klucza publicznego. Jak zauważono w kontekście szyfrowania kluczem prywatnym, żaden deterministyczny schemat szyfrowania nie może być bezpieczny pod kątem CPA. To samo dotyczy tutaj:

TWIERDZENIE 11.4 Żaden deterministyczny schemat szyfrowania klucza publicznego nie jest CPA-bezpieczne.

Ponieważ Twierdzenie 11.4 jest tak ważne, zasługuje na nieco szersze omówienie. Twierdzenie to nie jest „artefaktem” naszej definicji bezpieczeństwa ani wskazówką, że nasza definicja jest zbyt silna. Deterministyczne schematy szyfrowania klucza publicznego są podatne na praktyczne ataki w realistycznych scenariuszach i nigdy nie powinny być stosowane.

Powodem jest to, że schemat deterministyczny nie tylko pozwala przeciwnikowi określić, kiedy ta sama wiadomość zostanie wysłana dwukrotnie (jak w przypadku ustalenia klucza prywatnego), ale także pozwala przeciwnikowi odzyskać wiadomość z prawdopodobieństwem 1, jeśli zestaw możliwych szyfrowane wiadomości są niewielkie. Rozważmy na przykład profesora szyfrującego oceny uczniów. Tutaj podsłuchujący wie, że ocena każdego ucznia musi mieścić się w zakresie {A, B, C, D, F}. Jeśli profesor zastosuje deterministyczny schemat szyfrowania kluczem publicznym, podsłuchujący może szybko ustalić rzeczywistą ocenę dowolnego ucznia, szyfrując wszystkie możliwe oceny i porównując wynik z podanym zaszyfrowanym tekstem.

Chociaż powyższe twierdzenie wydaje się zwodniczo proste, przez długi czas wiele rzeczywistych systemów projektowano przy użyciu deterministycznego szyfrowania kluczem publicznego. Kiedy wprowadzono szyfrowanie kluczem publicznym, można śmiało powiedzieć, że znaczenie szyfrowania probabilistycznego nie było jeszcze w pełni uświadomione. Przełomowa praca Goldwassera i Micali, w której zaproponowano (coś równoważnego) definicji 11.2 i sformułowano twierdzenie 11.4, stanowiła punkt zwrotny w dziedzinie kryptografii. Nie należy niedoceniać znaczenia ugruntowania swojej intuicji w formalnej definicji i spojrzenia na sprawy we właściwy sposób po raz pierwszy – nawet jeśli z perspektywy czasu wydają się one proste.

11.2.2 Wiele szyfrowań

Podobnie jak w rozdziale 3, ważne jest zrozumienie skutków użycia tego samego klucza (w tym przypadku tego samego klucza publicznego) do szyfrowania wielu wiadomości. Moglibyśmy sformułować bezpieczeństwo w takim układzie, wysyłając przeciwnikowi dwie listy tekstów jawnych, jak w definicji 3.19. Jednakże z powodów omówionych w sekcji 3.4.2 zamiast tego zdecydowaliśmy się użyć definicji, w której atakujący ma dostęp do „lewej lub prawej” wyroczni $LRpk,b$, która po wprowadzeniu pary wiadomości o równej długości m_0, m_1 , oblicza szyfrogram $c = Encpk(mb)$ i zwraca c . Osoba atakująca może odpypytać tę wyrocznię tyle razy, ile chce, dlatego definicja ta modeluje bezpieczeństwo, gdy wiele (nieznanych) wiadomości jest szyfrowanych przy użyciu tego samego klucza publicznego.

Formalnie rozważmy następujący eksperyment zdefiniowany dla szyfrowania klucza publicznego: schemat $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ i przeciwnik A:

Eksperyment $LR\text{-Oracle PubKLR-cpa } (n): A, \Pi$

1. Uruchomiono $\text{Gen}(1^n)$ w celu uzyskania kluczy (pk, sk).
2. Wybierany jest bit jednolity $b \in \{0, 1\}$.
3. Przeciwnik A otrzymuje dane wejściowe pk i dostęp Oracle do $LRpk,b(\cdot, \cdot)$.
4. Przeciwnik A wysyła bit b .
5. Wynik eksperymentu definiuje się jako 1, jeśli $b = b$, i 0 w przeciwnym razie. Jeśli $\text{PubKLR-cpa } (n) = 1$, mówimy, że A się udało.

DEFINICJA 11.5 Schemat szyfrowania kluczem publicznym $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ charakteryzuje się nieroróżnialnymi wielokrotnymi szyfrowaniami , jeśli dla wszystkich probabilistycznych przeciwników A w czasie wielomianowym istnieje pomijalna funkcja negl taka, że:

$$\Pr[\text{PubKLR-cpa}(\Pi) = 1] \geq \frac{1}{2} + \text{zaniedbywanie}(n).$$

Pokażemy, że każdy schemat zabezpieczony CPA automatycznie ma wiele nieroróżnialnych szyfrowań; to znaczy, że w ustawieniu klucza publicznego bezpieczeństwo szyfrowania pojedynczej wiadomości oznacza bezpieczeństwo szyfrowania wielu wiadomości. Oznacza to, że możemy udowodnić bezpieczeństwo jakiegoś schematu w odniesieniu do definicji 11.2, która jest prostsza i łatwiejsza w obsłudze, i stwierdzić, że schemat spełnia definicję 11.5, pozornie silniejszą definicję, która dokładniej modeluje wykorzystanie szyfrowania klucza publicznego w świecie rzeczywistym . Poniżej przedstawiono dowód następującego twierdzenia.

TWIERDZENIE 11.6 Jeżeli schemat szyfrowania klucza publicznego Π jest zabezpieczony CPA, to posiada on także nieroróżnialne wielokrotne szyfrowania.

Stwierdzono analogiczny wynik w przypadku ustawienia klucza prywatnego, ale nie został on udowodniony, jako Twierdzenie 3.24.

Szyfrowanie wiadomości o dowolnej długości. Bezpośrednią konsekwencją Twierdzenia 11.6 jest to, że schemat szyfrowania kluczem publicznym z zabezpieczeniem CPA dla wiadomości o stałej długości implikuje schemat szyfrowania kluczem publicznym dla wiadomości o dowolnej długości, spełniający to samo pojęcie bezpieczeństwa. Zilustrujemy to w skrajnym przypadku, gdy oryginalny schemat szyfruje tylko wiadomości 1-bitowe. Powiedzmy, że $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ to schemat szyfrowania wiadomości jednobitowych. Możemy skonstruować nowy schemat $\Pi' = (\text{Gen}', \text{Enc}', \text{Dec}')$, który ma przestrzeń komunikatów $\{0, 1\}^n$, definiując Enc' w następujący sposób:

$$\text{Enc}' \text{pk}(m) = \text{Encpk}(m_1), \dots, \text{Encpk}(m_n), \quad (11.1)$$

gdzie $m = m_1 \cdots m_n$. (Algorytm deszyfrujący Dec' jest skonstruowany w oczywisty sposób.) Mamy:

Twierdzenie 11.7 Niech Π i Π' będą jak powyżej. Jeżeli Π jest zabezpieczone CPA, to Π' również.

Twierdzenie wynika z tego, że możemy postrzegać szyfrowanie wiadomości m przy użyciu Π' jako szyfrowanie t wiadomości (m_1, \dots, m_n) przy użyciu schematu Π .

Uwaga dotycząca terminologii. Wprowadziłmy trzy definicje bezpieczeństwa schematów szyfrowania klucza publicznego – szyfrowanie nieroróżnialne w obecności podsłuchującego, bezpieczeństwo CPA i nieroróżnialne wielokrotne szyfrowanie – które są równoważne. Zgodnie z konwencją stosowaną w literaturze kryptograficznej, będziemy po prostu używać terminu „bezpieczeństwo CPA” w odniesieniu do schematów spełniających te pojęcia bezpieczeństwa.

*Dowód twierdzenia 11.6

Dowód twierdzenia 11.6 jest raczej skomplikowany. Dlatego zanim przejdziemy do szczegółów, dajemy trochę intuicji. Na potrzeby tej intuicyjnej dyskusji zakładamy dla uproszczenia, że A wykonuje tylko dwa wywołania do wyroczni LR w eksperymencie PubKLR-cpa_{A,Π}(n). (W pełnym dowodzie liczba wywołań jest dowolna.)

Napraw dowolnego przeciwnika ppt A i schemat szyfrowania klucza publicznego z zabezpieczeniem CPA Π i rozważ eksperiment PubKLR-cpa_{A,Π}2(n), w którym A może wykonać tylko dwa zapytania do wyroczni LR. Oznacz zapytania zadane przez A wyroczni za pomocą (m_{1,0}, m_{1,1}) i (m_{2,0}, m_{2,1}); zauważ, że druga para wiadomości może zależeć od pierwszego zaszyfrowanego tekstu uzyskanego przez A od wyroczni. W eksperymencie A otrzymuje albo parę szyfrogramów (Encpk(m_{1,0}), Encpk(m_{2,0})) (jeśli b = 0), albo parę szyfrogramów (Encpk(m_{1,1}), Encpk(m_{2,1})) (jeśli b = 1). Zapisujemy A(pk, Encpk(m_{1,0}), Encpk(m_{2,0})), aby oznaczyć wynik A w pierwszym przypadku i analogicznie w drugim. oznaczają rozkład

Niech C₀ par szyfrogramów w pierwszym przypadku, a C rozkład par szyfrogramów₁ w drugim przypadku. Aby pokazać, że definicja 11.5 jest spełniona (dla PubKLR-cpa_{A,Π}2), musimy udowodnić, że A nie może rozróżnić pomiędzy otrzymaniem pary szyfrogramów rozdzielonych według C₀ lub pary szyfrogramów rozdzielonych według C. Oznacza to, że potrzebujemy udowodnić, że istnieje funkcja pomijalna taka, że

$$\Pr[A(\text{pk}, \text{Encpk}(m_{1,0}), \text{Encpk}(m_{2,0})) = 1]$$

$$\Pr[A(\text{pk}, \text{Encpk}(m_{1,1}), \text{Encpk}(m_{2,1})) = 1] \quad \text{negl}(n). \quad (11.2)$$

(Jest to równoważne definicji 11.5 z tego samego powodu, dla którego definicja 3.9 jest równoważna definicji 3.8.) Aby to udowodnić, pokażemy, że

- Bezpieczeństwo CPA Π oznacza, że A nie może rozróżnić przypadku, gdy otrzyma parę szyfrogramów rozdzielonych według C₀ lub parę szyfrogramów (Encpk(m_{1,0}), Encpk(m_{2,1})), co odpowiada szyfrowaniu pierwszej wiadomości w pierwszym zapytaniu Oracle A i drugiej wiadomości w drugim zapytaniu Oracle A. (Chociaż nie może to mieć miejsca w PubKLR-cpa_{A,Π}2(n), wciąż możemy zapytać, jakie byłoby zachowanie A, gdyby A, Π dostał taką parę szyfrogramów.) Niech C tworzy pary w tym drugim C_0 oznaczającą rozkład tekstu zaszyfrowanego przypadku.

- Podobnie bezpieczeństwo CPA Π oznacza, że A nie może rozróżnić przypadku, gdy otrzyma parę szyfrogramów rozdzielonych według C₀₁ lub parę szyfrogramów rozdzielonych według C₁.

Z powyższego wynika, że A nie potrafi rozróżnić rozkładów C₀ ani rozkładów C₁ i C₀₁. Dochodzimy do wniosku (używając prostej algebra) C₀ i C₁.

że A nie potrafi rozróżnić rozkładów C₀

Sednem dowodu jest zatem wykazanie, że A nie może rozróżnić pomiędzy otrzymaniem pary szyfrogramów rozdzielonych według C₀, a parą szyfrów

szyfrogramy dystrybuowane zgodnie z C 01. (Drugi przypadek następuje podobnie.) Oznacza to, że chcemy pokazać, że istnieje pomijalna funkcja negl dla której

$$\begin{aligned} \Pr[A(\text{pk}, \text{Encpk}(m1,0), \text{Encpk}(m2,0)) = 1] \\ \Pr[A(\text{pk}, \text{Encpk}(m1,0), \text{Encpk}(m2,1)) = 1] \quad \text{negl}(n). \end{aligned} \quad (11.3)$$

Należy zauważyć, że jedyna różnica między danymi wejściowymi przeciwnika A w każdym przypadku dotyczy drugiego elementu. Intuicyjnie nieroróżnialność wynika z przypadku pojedynczego komunikatu, ponieważ A może samodzielnie wygenerować $\text{Encpk}(m1,0)$. Formalnie rozważmy następującego przeciwnika ppt A biorącego udział w eksperymencie $\text{PubKeav A ,}\Pi(n)$:

Przeciwnik A:

1. Na wejściu pk przeciwnik A wykonuje $A(\text{pk})$ jako podprogram.
2. Kiedy A wykonuje swoje pierwsze zapytanie $(m1,0, m1,1)$ do wyroczni LR , A oblicza $c1 = \text{Encpk}(m1,0)$ i zwraca $c1$ do A jako odpowiedź z wyroczni.
3. Kiedy A wykonuje drugie zapytanie $(m2,0, m2,1)$ do LR or-acle, A wysyła $(m2,0, m2,1)$ i otrzymuje z powrotem zaszyfrowany tekst wyzwania $c2$. Jest to zwracane do A jako odpowiedź z wyroczni LR .
4. A wysyła bit b na wyjście A.

Patrząc na eksperyment $\text{PubKeav A ,}\Pi(n)$, widzimy, że gdy $b = 0$, wówczas szyfrogram wyzwania $c2$ jest obliczany jako $\text{Encpk}(m2,0)$. Zatem,

$$\Pr[A(\text{Encpk}(m2,0)) = 1] = \Pr[A(\text{Encpk}(m1,0), \text{Encpk}(m2,0)) = 1]. \quad (11.4)$$

(Pomijamy wyraźną wzmiankę o pk, aby zaoszczędzić miejsce.) Natomiast, gdy $b = 1$ w eksperymencie $\text{PubKeav A ,}\Pi(n)$, wówczas $c2$ jest obliczane jako $\text{Encpk}(m2,1)$ i tak

$$\Pr[A(\text{Encpk}(m2,1)) = 1] = \Pr[A(\text{Encpk}(m1,0), \text{Encpk}(m2,1)) = 1]. \quad (11.5)$$

Bezpieczeństwo CPA Π oznacza, że istnieje nieistotna funkcja zaniedbująca taka, że

$$|\Pr[A(\text{Encpk}(m2,0)) = 1] - \Pr[A(\text{Encpk}(m2,1)) = 1]| \leq \text{zaniedb}(n).$$

To, wraz z równaniami (11.4) i (11.5), daje równanie (11.3).

Niemal dokładnie w ten sam sposób możemy udowodnić, że:

$$\begin{aligned} \Pr[A(\text{pk}, \text{Encpk}(m1,0), \text{Encpk}(m2,1)) = 1] \\ \Pr[A(\text{pk}, \text{Encpk}(m1,1), \text{Encpk}(m2,1)) = 1] \quad \text{negl}(n). \end{aligned} \quad (11.6)$$

Równanie (11.2) następuje poprzez połączenie równań (11.3) i (11.6).

Główną komplikacją, która pojawia się w ogólnym przypadku, jest to, że liczba zapytań do wyroczni LR nie jest już stała, lecz może być dowolna

wielomian n. W dowodzie formalnym rozwiązuje się to za pomocą argumentu hybrydowego. (Argumenty hybrydowe zostały użyte także w rozdziale 7.)

DOWÓD (Twierdzenia 11.6) Niech Π będzie schematem szyfrowania klucza publicznego z zabezpieczeniem CPA, a A dowolnym przeciwnikiem ppt w eksperymencie PubKLR-cpa(n).
 Niech $t = t(n)$ będzie wielomianem górnej granicy liczby zapytań kierowanych przez A do wyroczni LR i załóż bez utraty ogólności, że A zawsze zadaje wyroczni dokładnie tyle razy. Dla danego klucza publicznego pk i $0 \leq t$, niech LRI oznacza wyrocznię, która na wejściu (m_0, m_1) zwraca $Encpk(m_0)$ dla pierwszego otrzymanego zapytania i zwraca $Encpk(m_1)$ dla następnego t i zapytania, które otrzymuje. (Oznacza to, że w przypadku pierwszego zapytania i pierwsza wiadomość z pary wejściowej jest szyfrowana, a w odpowiedzi na pozostałe zapytania szyfrowana jest druga wiadomość z pary wejściowej). Podkreślamy, że każde szyfrowanie jest obliczane przy użyciu jednolitej, niezależnej losowości. Używając tego zapisu, mamy

$$\Pr_{A, \Pi}[\text{PubKLR-cpa}(n) = 1] = \frac{1}{2} \cdot \Pr[A \xrightarrow{\text{LRt}} pk(pk) = 0] + \frac{1}{2} \cdot \Pr[A \xrightarrow{\text{LR0}} pk(pk) = 1]$$

ponieważ z punktu widzenia A (który wykonuje dokładnie t zapytań) Oracle LRt jest równojszczelne dla A. Aby udowodnić spełnienie definicję 11.5, pokażemy, że dla dowolnego ppt A istnieje pomijalna funkcja negl taka, że

$$\Pr[A \xrightarrow{\text{LRt}} pk(pk) = 1] - \Pr[A \xrightarrow{\text{LR0}} pk(pk) = 1] = \text{negl}(n). \quad (11.7)$$

(Tak jak poprzednio, jest to równoważne definicji 11.5 z tego samego powodu, dla którego definicja 3.9 jest równoważna definicji 3.8.)

Rozważmy następującego przeciwnika ppt A, który podsłuchuje szyfrowanie pojedynczej wiadomości:

Przeciwnik A:

1. A, mając dane pk, wybiera jednolity indeks $i \in \{1, \dots, T\}$.
2. A uruchamia $A(pk)$, odpowiadając na swoje j-te zapytanie wyroczni $(mj, 0, mj, 1)$ jako następująco:
 - (a) Dla $j < i$ przeciwnik A oblicza $c_j = Encpk(mj, 0)$ i zwraca c_j do A jako odpowiedź swojej wyroczni.
 - (b) Dla $j = i$, przeciwnik A wyrowadza $(mj, 0, mj, 1)$ i otrzymuje z powrotem zaszyfrowany tekst wyzwania c_j . Jest to zwracane do A jako odpowiedź od jego wyroczni.
- (c) Dla $j > i$ przeciwnik A oblicza $c_j = Encpk(mj, 1)$ i zwraca c_j do A jako odpowiedź swojej wyroczni.
3. A wyrowadza bit b, który jest wysyłany przez A.

Rozważ eksperyment PubKeav, $A, \Pi(n)$. Naprawianie pewnego wyboru $i = i_{\text{ja}}$, zauważ to jeśli ci i jest szyfrowaniem mi $\text{pk}(pk)$, to interakcja A z jego wyrocznią jest identyczna z interakcją z wyrocznią LR^i . Zatem,

$$\begin{aligned} \Pr[A \text{ wyjście } 1 \mid b = 0] &= \Pr_{\substack{i=1 \\ \text{ja}}} [i = i_{\text{ja}}] \cdot \Pr[A \text{ wyjście } 1 \mid b = 0 \quad \text{ja} = \text{ja}] \\ &= \frac{1}{T} \cdot \Pr_{\substack{\text{ja}=1 \\ T}} [A \xrightarrow{\text{pk}(pk)} 1]. \end{aligned}$$

Z drugiej strony, jeśli ci i jest szyfrowaniem mi $\text{pk}(pk)$, to interakcja A z jego wyrocznią jest identyczna z interakcją z wyrocznią LR^i i tak $\text{pk}(pk) = 1$.

$$\begin{aligned} \Pr[A \text{ wyjście } 1 \mid b = 1] &= \Pr_{\substack{i=1 \\ \text{ja}}} [i = i_{\text{ja}}] \cdot \Pr[A \text{ wyjście } 1 \mid b = 1 \quad \text{ja} = \text{ja}] \\ &= \frac{1}{T} \cdot \Pr_{\substack{\text{ja}=1 \\ T}} [A \xrightarrow{\text{pk}(pk)} 1] \\ &= \frac{1}{T} \cdot \Pr_{\substack{i=0 \\ t-1}} [A \xrightarrow{\text{pk}(pk)} 1], \end{aligned}$$

gdzie trzecią równość uzyskuje się po prostu przez przesunięcie wskaźników sumowania.

Ponieważ A przebiega w czasie wielomianowym, założenie, że Π jest bezpieczne dla CPA, oznacza, że istnieje pomijalna funkcja negl taka, że

$$\Pr[A \text{ wyjście } 1 \mid b = 0] = \Pr[A \text{ wyjście } 1 \mid b = 1] = \text{negl}(n).$$

Ale to oznacza, że

$$\begin{aligned} \text{zaniedb}(n) &= \Pr_{\substack{\text{ja}=1 \\ t}} [A \xrightarrow{\text{pk}(pk)} 1] - \Pr_{\substack{\text{ja}=0 \\ T}} [A \xrightarrow{\text{pk}(pk)} 1] \\ &= \frac{1}{T} \cdot \Pr[A \xrightarrow{\text{pk}(pk)} 1] - \Pr[A \xrightarrow{\text{pk}(pk)} 0] = \text{negl}(n), \end{aligned}$$

ponieważ wszystkie wyrazy w każdym podsumowaniu z wyjątkiem jednego znoszą się. Dochodzimy do wniosku, że

$$\Pr[A \xrightarrow{\text{pk}(pk)} 1] = \Pr[A \xrightarrow{\text{pk}(pk)} 0] = \text{negl}(n).$$

Ponieważ t jest wielomianem, funkcja $t \cdot \text{negl}(n)$ jest pomijalna. Ponieważ A był arbitralnym przeciwnikiem ppt, pokazuje to, że równanie (11.7) zachodzi, co kończy dowód, że Π ma nieroróżnialne wielokrotne szyfrowania. ■

11.2.3 Zabezpieczenie przed atakami z wykorzystaniem wybranego tekstu zaszyfrowanego

Ataki z użyciem wybranego tekstu zaszyfrowanego, podczas których przeciwnik jest w stanie odszyfrować dowolny wybrany przez siebie tekst zaszyfrowany (z jednym ograniczeniem technicznym opisanym poniżej), stanowią problem w przypadku ustawienia klucza publicznego, tak samo jak w przypadku klucza prywatnego ustawienie. W rzeczywistości są one prawdopodobnie większym problemem w przypadku ustawienia klucza publicznego, ponieważ tam odbiorca spodziewa się otrzymać zaszyfrowane teksty od wielu nadawców, którzy prawdopodobnie są z góry nieznani, podczas gdy odbiorca przy ustawieniu klucza prywatnego zamierza komunikować się tylko z jednym, znany nadawca używający dowolnego tajnego klucza.

Załóżmy, że podsłuchujący A obserwuje tekst zaszyfrowany c wysłany przez nadawcę S do odbiorcy R. Ogólnie rzeczą biorąc, w ustawieniu klucza publicznego istnieją dwie klasy ataków z użyciem wybranego tekstu zaszyfrowanego:

- A może wysłać zmodyfikowany tekst zaszyfrowany c do R w imieniu S. (Na przykład w kontekście zaszyfrowanej wiadomości e-mail A może utworzyć zaszyfrowaną wiadomość e-mail c i sfalszować pole „Od” tak, aby wyglądało na e -mail pochodzący od S.) W tym przypadku, chociaż jest mało prawdopodobne, aby A był w stanie uzyskać całe odszyfrowanie m z c, możliwe byłoby, aby A wywnioskował pewne informacje o m na podstawie późniejszego zachowania R.

Na podstawie tych informacji A może dowiedzieć się czegoś o oryginalnej wiadomości, m.in.

- A może wysłać zmodyfikowany tekst zaszyfrowany c do R we własnym imieniu. W tym przypadku A może uzyskać całe odszyfrowanie m z c, jeśli R odpowie bezpośrednio A. Nawet jeśli A nie dowie się niczego o m, ta zmodyfikowana wiadomość może mieć znany związek z oryginalną wiadomością m, która może zostać wykorzystana przez A; zobacz przykład trzeciego scenariusza poniżej.

Druga klasa ataków jest specyficzna dla kontekstu szyfrowania klucza publicznego i nie ma odpowiednika w przypadku ustawienia klucza prywatnego.

Nietrudno wskazać kilka realistycznych scenariuszy ilustrujących powyższe typy ataków:

Scenariusz 1. Założmy, że użytkownik S loguje się na swoje konto bankowe, wysyłając do swojego banku zaszyfrowane hasło pw połączone ze znacznikiem czasu. Założmy dalej, że bank wysyła dwa rodzaje komunikatów o błędach: zwraca „nieprawidłowe hasło”, jeśli zaszyfrowane hasło nie pasuje do zapisanego hasła S, oraz „nieprawidłowy znacznik czasu”, jeśli hasło jest poprawne, ale znaczek czasu nie.

Jeśli przeciwnik uzyska zaszyfrowany tekst c wysłany przez S do banku, może teraz przeprowadzić atak z użyciem wybranego tekstu zaszyfrowanego, wysyłając zaszyfrowany tekst c do banku w imieniu S i obserwując wynikające z tego komunikaty o błędach. (Jest to podobne do ataku padding-oracle, który widzieliśmy w sekcji 3.7.2.) W niektórych przypadkach ta informacja może wystarczyć, aby umożliwić przeciwnikowi ustalenie całego hasła użytkownika.

Scenariusz 2. Założymy, że S wysyła zaszyfrowaną wiadomość e-mail c do R, a wiadomość ta jest obserwowana przez A. Jeśli A wysyła we własnym imieniu zaszyfrowaną wiadomość e-mail c do R, wówczas R może na nią odpowiedzieć -mail i zacytuje odszyfrowany tekst m odpowiadający c. W tym przypadku R zasadniczo działa jako wyrocznia deszyfrująca dla A i może potencjalnie odszyfrować każdy tekst zaszyfrowany, który A mu wysyła.

Scenariusz 3. Kwestią ściśle związaną z bezpieczeństwem wybranego tekstu zaszyfrowanego jest potencjalna plastyczność zaszyfrowanych tekstów. Ponieważ formalna definicja jest dość skomplikowana, nie podajemy jej tutaj, a jedynie intuicyjną ideę.

Schemat jest plastyczny, jeśli ma następującą właściwość: biorąc pod uwagę szyfrowanie c nieznanej wiadomości m, możliwe jest wymyślenie tekstu zaszyfrowanego c będącego zaszyfrowaniem wiadomości m, która jest w jakiś znany sposób powiązana z m.

Na przykład, być może biorąc pod uwagę szyfrowanie m, możliwe jest skonstruowanie szyfrowania 2m. (Naturalne przykłady schematów bezpiecznych według CPA z tą i podobnymi właściwościami zobaczymy później; patrz sekcja 13.2.3.)

Teraz wyobraź sobie, że R prowadzi aukcję, podczas której dwie strony S i A składają swoje oferty, szyfrując je przy użyciu klucza publicznego R. Jeśli zostanie zastosowany plastyczny schemat szyfrowania, może się zdarzyć, że przeciwnik A zawsze złoży najwyższą ofertę (bez licytowania maksimum), przeprowadzając następujący atak: poczekaj, aż S wyśle zaszyfrowany tekst c odpowiadający jego ofercie m (nieznanej A); następnie wyślij szyfrogram c odpowiadający ofercie $m = 2m$. Należy zauważyć, że m (w tym przypadku m) pozostają nieznane A do czasu, gdy R ogłosí wyniki, zatem możliwość takiego ataku nie przeczy faktowi, że schemat szyfrowania jest zabezpieczony CPA. Z drugiej strony, schematy zabezpieczone przez CCA można wykazać, że są nieplastyczne, co oznacza, że nie są podatne na takie ataki.

Definicja. Bezpieczeństwo przed atakami z wykorzystaniem wybranego tekstu zaszyfrowanego definiuje się poprzez odpowiednią modyfikację analogicznej definicji z ustawieniem klucza prywatnego (Definicja 3.33). Biorąc pod uwagę schemat szyfrowania kluczem publicznym Π i przeciwnika A, rozważ następujący eksperyment:

Eksperyment nieroróżnialności CCA $\text{PubKcca } A, \Pi(n)$:

1. Uruchomiono $\text{Gen}(1n)$ w celu uzyskania kluczy (pk, sk).
2. Przeciwnik A otrzymuje pk i dostęp do deszyfrowania lub akle $\text{Decsk}(\cdot)$. Wysyła parę komunikatów m_0, m_1 o tej samej długości. (Te wiadomości muszą znajdować się w przestrzeni wiadomości powiązanej z pk.)
3. Wybierany jest bit jednolity $b \in \{0, 1\}$, a następnie obliczany jest szyfrogram c $\text{Encpk}(mb)$ i przekazywany A.
4. A kontynuuje interakcję z wyrocznią deszyfrującą, ale nie może zażądać odszyfrowania samego c. Na koniec A wyprowadza trochę b.
5. Wynik eksperymentu definiuje się jako 1, jeśli $b = b$ i 0 w przeciwnym razie.

DEFINICJA 11.8 Schemat szyfrowania kluczem publicznym $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ ma nieroróżnicalne szyfrowanie w przypadku ataku wybranym tekstem zaszyfrowanym (lub jest bezpieczny CCA), jeśli dla wszystkich probabilistycznych przeciwników A w czasie wielomianowym istnieje nieistotna funkcja negl taka, że

$$\Pr[\text{PubKcca } A, \Pi(n) = 1] \xrightarrow[2]{\text{zaniebywanie (n).}} 1$$

Naturalny odpowiednik Twierdzenia 11.6 dotyczy także bezpieczeństwa CCA. Oznacza to, że jeśli schemat ma nieroróżnicalne szyfrowania w przypadku ataku z użyciem wybranego tekstu zaszyfrowanego, to w przypadku ataku z wybranym tekstem zaszyfrowanym ma nieroróżnicalne wielokrotne szyfrowania, jeśli jest to odpowiednio zdefiniowane. Co ciekawe, analogia do zastrzeżenia 11.7 nie dotyczy zabezpieczenia CCA.

Podobnie jak w definicji 3.33, musimy uniemożliwić atakującemu przesłanie zaszyfrowanego tekstu wyzwania c do wyroczni deszyfrującej, aby definicja była możliwa do osiągnięcia. Ale to ograniczenie nie pozbawia definicji znaczenia i w szczególności dla każdego z trzech podanych wcześniej scenariuszy motywacyjnych można argumentować, że ustwienie $c = c$ nie przynosi korzyści atakującemu:

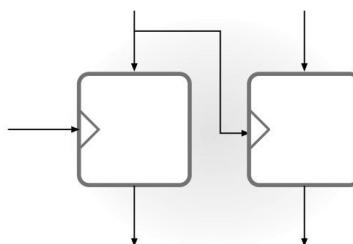
- W pierwszym scenariuszu obejmującym logowanie w oparciu o hasło, osoba atakująca nie dowiaduje się niczego o haśle S , odtwarzając c do banku, ponieważ w tym przypadku już wie, że nie zostanie wygenerowany żaden komunikat o błędzie.
- W drugim scenariuszu dotyczącym zaszyfrowanej wiadomości e-mail wysłanie $c = c$ do odbiorcy prawdopodobnie wzbudziłoby w nim podejrzenia, w związku z czym w ogóle odmówiłby odpowiedzi.
- W ostatecznym scenariuszu obejmującym aukcję R może z łatwością wykryć oszustwo, jeśli zaszyfrowana oferta przeciwnika jest identyczna z zaszyfrowaną ofertą drugiej strony. Tak czy inaczej, w tym przypadku wszystko, co atakujący osiągnie poprzez ponowne rozegranie c , to złożenie tej samej oferty, co strona uczciwa.

11.3 Szyfrowanie hybrydowe i paradygmat KEM/DEM

Twierdzenie 11.7 pokazuje, że dowolny schemat szyfrowania kluczem publicznym zabezpieczonym CPA dla wiadomości bitowych może zostać wykorzystany do uzyskania schematu szyfrowania kluczem publicznym zabezpieczonym CPA dla wiadomości o dowolnej długości. Szyfrowanie wiadomości $-bit$ przy użyciu tego podejścia wymaga $y = /$ wywołań oryginalnego schematu szyfrowania, co oznacza, że zarówno obliczenia, jak i długość tekstu zaszyfrowanego są zwiększone o współczynnik mnożenia y w stosunku do podstawowego schematu.

Można osiągnąć lepsze rezultaty, stosując szyfrowanie kluczem prywatnym w połączeniu z szyfrowaniem kluczem publicznym. Poprawia to wydajność, ponieważ szyfrowanie kluczem prywatnym jest znacznie szybsze niż szyfrowanie kluczem publicznym i poprawia szerokość pasma, ponieważ schematy klucza prywatnego mają mniejszą ekspansję tekstu zaszyfrowanego. Powstała kombinacja nazywana jest szyfrowaniem hybrydowym i jest szeroko stosowana w

ćwiczyć. Podstawową ideą jest użycie szyfrowania kluczem publicznym w celu uzyskania klucza współdzielonego k , a następnie zaszyfrowanie wiadomości m przy użyciu schematu szyfrowania klucza prywatnego i klucza k . Odbiorca używa swojego długoterminowego (asymetrycznego) klucza prywatnego do uzyskania k , a następnie wykorzystuje deszyfrowanie klucza prywatnego (za pomocą klucza k), aby odzyskać oryginalną wiadomość. Podkreślamy, że chociaż jako składnik wykorzystywane jest szyfrowanie kluczem prywatnym, jest to pełnoprawny schemat szyfrowania kluczem publicznym ze względu na fakt, że nadawca i odbiorca nie dzielą z góry żadnego tajnego klucza.



RYSUNEK 11.1: Szyfrowanie hybrydowe. Enc oznacza schemat szyfrowania klucza publicznego, natomiast Enc to schemat szyfrowania klucza prywatnego.

W bezpośredniej implementacji tego pomysłu (patrz rysunek 11.1) nadawca podzieliłby k poprzez (1) wybranie jednolitej wartości k , a następnie (2) zaszyfrowanie k przy użyciu schematu szyfrowania klucza publicznego. Bardziej bezpośrednie podejście polega na użyciu prymitywnego klucza publicznego zwanego mechanizmem enkapsulacji klucza (KEM), aby osiągnąć oba te cele „za jednym razem”. Jest to korzystne zarówno z koncepcyjnego punktu widzenia, jak i pod względem wydajności, jak zobaczymy później.

KEM ma trzy algorytmy podobne w duchu do schematu szyfrowania klucza publicznego. Tak jak poprzednio, algorytm generowania kluczy Gen służy do generowania pary kluczy publicznych i prywatnych. Zamiast szyfrowania mamy teraz algorytm enkapsulacji Encaps, który jako dane wejściowe pobiera tylko klucz publiczny (bez wiadomości) i generuje zaszyfrowany tekst c wraz z kluczem k . Odpowiedni algorytm dekapsulacji Decaps jest uruchamiany przez odbiorcę w celu odzyskania k z zaszyfrowanego tekstu c przy użyciu klucza prywatnego. Formalnie:

DEFINICJA 11.9 Mechanizm enkapsulacji klucza (KEM) jest krotką probabilistycznych algorytmów czasu wielomianowego (Gen, Encaps, Decaps), takich że:

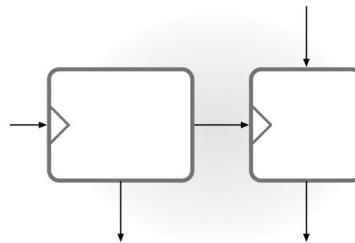
1. Algorytm generowania klucza Gen przyjmuje jako dane wejściowe parametr bezpieczeństwa n i wyprowadza parę kluczy publiczny/prywatny (pk, sk). Zakładamy, że pk ter 1 i sk długość co najmniej n i że n można wyznaczyć z pk .
2. Algorytm enkapsulacji Encaps przyjmuje na wejściu klucz publiczny pk i parametr bezpieczeństwa $1/n$. Wysyła tekst zaszyfrowany c i klucz $k \in \{0, 1\}^n$ gdzie jest długością klucza. Zapisujemy to jako $(c, k) = \text{Encaps}(pk, 1/n)$.

3. Deterministyczny algorytm dekapsulacji Decaps przyjmuje jako dane wejściowe klucz prywatny sk i tekst zaszyfrowany c, a na wyjściu otrzymuje klucz k lub specjalny symbol ozaczający awarię. Zapisujemy to jako $k := \text{Decapssk}(c)$.

Wymagane jest, aby z prawie znikomym prawdopodobieństwem ponad (sk, pk) wyjściem przez $\text{Gen}(1n)$, jeśli $\text{Encapspk}(1n)$ dało wynik (c, k) , to $\text{Decapssk}(c)$ dał wynik k .

W definicji zakładamy dla uproszczenia, że Encaps zawsze wprowadza (zaszyfrowany tekst c i) klucz o określonej długości (n). Można również rozważyć bardziej ogólną definicję, w której Encaps przyjmuje 1 jako dodatkowe wejście i wprowadza klucz długości.

Każdy schemat szyfrowania klucza publicznego w prosty sposób daje KEM, wybierając losowy klucz k i szyfrując go. Jak się jednak okaże, dedykowane konstrukcje KEM-ów mogą być bardziej wydajne.



RYSUNEK 11.2: Szyfrowanie hybrydowe przy użyciu metody KEM/DEM.

Używając KEM (o długości klucza n), możemy wdrożyć szyfrowanie hybrydowe, jak na rysunku 11.2. Nadawca uruchamia $\text{Encapspk}(1n)$, aby uzyskać c wraz z kluczem k ; następnie używa schematu szyfrowania klucza prywatnego do szyfrowania swojej wiadomości m , używając k jako klucza. W tym kontekście schemat szyfrowania klucza prywatnego z oczywistych powodów nazywany jest mechanizmem enkapsulacji danych (DEM). Zaszyfrowany tekst wysłany do odbiorcy zawiera zarówno c , jak i zaszyfrowany tekst c ze schematu klucza prywatnego. Konstrukcja 11.10 podaje formalną specyfikację.

Jaka jest wydajność otrzymanego hybrydowego schematu szyfrowania Π_{hy} ? Dla pewnej ustalonej wartości n niech α oznacza koszt enkapsulacji n -bitowego klucza przy użyciu Encaps i niech β oznacza koszt (na bit zwykłego tekstu) szyfrowania przy użyciu Enc. Założymy, że $|m| > n$, co jest interesującym przypadkiem. Zatem koszt zaszyfrowania wiadomości m przy użyciu Π_{hy} wynosi w przeliczeniu na bit zwykłego tekstu

$$\frac{\alpha + \beta \cdot |m|}{|m|} = \frac{\alpha}{|m|} + \beta, \quad (11.8)$$

który zbliża się do β przez wystarczająco długi czas m . Zatem w przypadku bardzo długich wiadomości koszt za bit ponoszony przez schemat szyfrowania klucza publicznego Π_{hy} wynosi

BUDOWA 11.10

Niech $\Pi = (\text{Gen}, \text{Encaps}, \text{Decaps})$ będzie KEM o długości klucza n i niech $\Pi = (\text{Gen}, \text{Dec})$ będzie schematem szyfrowania klucza prywatnego. Skonstruj schemat szyfrowania kluczem publicznym $\text{Enc } \Pi_{\text{hy}} = (\text{Gen}_{\text{hy}}, \text{Enchy}, \text{Dechy})$ w następujący sposób:

- Gen_{hy}: na wejściu $1n$ uruchom $\text{Gen}(1n)$ i użij kluczy publicznych i prywatnych (pk , sk), które zostaną wyprowadzone.
- Enchy: po wprowadzeniu klucza publicznego pk i wiadomości $m \in \{0, 1\}$ wykonaj:
 1. Oblicz $(c, k) = \text{Encaps}_{\text{pk}}(1n)$.
 2. Oblicz $c = \text{Enc}_k(m)$.
 3. Wyprowadź zaszyfrowany tekst c, c .
- Dechy: po wprowadzeniu klucza prywatnego sk i tekstu zaszyfrowanego c, c wykonaj:
 1. Oblicz $k := \text{Decaps}_{\text{sk}}(c)$.
 2. Wypisz komunikat $m := \text{Dec}_k(c)$.

Szyfrowanie hybrydowe z wykorzystaniem paradygmatu KEM/DEM.

taki sam jak koszt za bit schematu klucza prywatnego Π . Szyfrowanie hybrydowe pozwala zatem osiągnąć funkcjonalność szyfrowania kluczem publicznym przy wydajności szyfrowania kluczem prywatnym, przynajmniej w przypadku wystarczająco długich wiadomości.

Podobne obliczenia można zastosować do pomiaru wpływu szyfrowania hybrydowego na długość tekstu zaszyfrowanego. Dla pewnej ustalonej wartości n , niech L oznacza długość tekstu zaszyfrowanego wyprowadzanego przez Encaps i powiedzmy, że szyfrowanie wiadomości kluczem prywatnym m przy użyciu Enc daje w wyniku szyfrogram o długości $n + |m|$ (można to osiągnąć stosując jeden z trybów szyfrowania omówionych w sekcji 3.6; w rzeczywistości możliwa jest nawet długość tekstu zaszyfrowanego $|m|$, ponieważ, jak zobaczymy, Π nie musi być zabezpieczone CPA). Wtedy całkowita długość tekstu zaszyfrowanego w schemacie Π_{hy} wynosi

$$L + n + |m|. \quad (11,9)$$

W przeciwnieństwie do tego, gdy stosuje się szyfrowanie blok po bloku, jak w równaniu (11.1) i zakładając, że szyfrowanie klucza publicznego n -bitowej wiadomości przy użyciu Enc daje w wyniku tekst zaszyfrowany o długości L , szyfrowanie wiadomości m skutkowałoby szyfrogram o długości $L \cdot |m|/n$. Długość tekstu zaszyfrowanego podana w równaniu (11.9) stanowi znaczną poprawę dla wystarczająco długiego m .

Możemy posłużyć się przybliżonymi szacunkami, aby zrozumieć, co powyższe wyniki oznaczają w praktyce. (Podkreślamy, że liczby te mają jedynie dać czytelnikowi poczucie poprawy; realistyczne wartości zależą od wielu czynników.) Typowa wartość długości klucza k może wynosić $n = 128$.

Co więcej, „natywny” schemat szyfrowania kluczem publicznym może generować 256-bitowy tekst zaszyfrowany podczas szyfrowania wiadomości 128-bitowych; założymy, że KEM ma szyfrogramy o tej samej długości podczas enkapsulacji 128-bitowego klucza. Pozwalając, aby a , jak poprzednio, oznaczało koszt obliczeniowy szyfrowania/hermetyzowania klucza publicznego 128-bitowego klucza, widzimy, że szyfrowanie blok po bloku, jak w równaniu (11.1),

zaszyfruj wiadomość o rozmiarze 1 MB (= 106 bitów) przy koszcie obliczeniowym $\alpha \cdot 106/128 \cdot 7800 \cdot \alpha$, a tekst zaszyfrowany będzie miał długość 2 MB. Porównaj to z wydajnością szyfrowania hybrydowego. Zakładając, że β , jak poprzednio, oznacza koszt obliczeniowy szyfrowania klucza prywatnego na bit, rozsądnym przybliżeniem jest $\beta = \alpha/105$.

Korzystając z równania (11.8), widzimy, że całkowity koszt obliczeniowy szyfrowania hybrydowego dla wiadomości o wielkości 1 Mb wynosi

$$\alpha + 106 \cdot \frac{\alpha}{105} = 11 \frac{\alpha}{105},$$

a szyfrogram byłby tylko nieco dłuższy niż 1 MB. Zatem szyfrowanie hybrydowe poprawia w tym przypadku wydajność obliczeniową 700-krotnie, a długość tekstu zaszyfrowanego 2-krotnie.

Pozostaje przeanalizować bezpieczeństwo Π_{hy} . Zależy to oczywiście od bezpieczeństwa podstawowych komponentów Π i Π . W kolejnych rozdziałach definiujemy pojęcia bezpieczeństwa CPA i bezpieczeństwa CCA dla KEM oraz pokazujemy:

- Jeśli Π jest KEM zabezpieczonym CPA, a schemat klucza prywatnego Π ma szyfrowanie nieroźnialne w obecności osoby podsłuchującej, wówczas Π_{hy} jest schematem szyfrowania klucza publicznego zabezpieczonego CPA. Należy zauważyć, że wystarczy, aby Π spełniał słabszą definicję bezpieczeństwa – co, jak pamiętamy, nie implikuje bezpieczeństwa CPA w ustawieniu klucza prywatnego – aby schemat hybrydowy Π_{hy} był bezpieczny CPA. Intuicyjnie powodem jest to, że za każdym razem, gdy szyfrowana jest nowa wiadomość, wybierany jest świeży, jednolity klucz k . Ponieważ każdy klucz k jest używany tylko raz, dla bezpieczeństwa schematu hybrydowego Π_{hy} wystarcza nieroźnialność pojedynczego szyfrowania Π . Oznacza to, że wystarczy podstawić szyfrowanie kluczem prywatnym przy użyciu generatora pseudolosowego (lub szyfru strumieniowego), jak w konstrukcji 3.17.
- Jeśli Π jest KEM zabezpieczonym przez CCA i Π jest schematem szyfrowania klucza prywatnego zabezpieczonego przez CCA, wówczas Π_{hy} jest schematem szyfrowania klucza publicznego zabezpieczonego przez CCA.

11.3.1 Bezpieczeństwo CPA

Dla uproszczenia zakładamy w tej i następnej sekcji KEM o długości klucza n . Pojęcie zabezpieczenia CPA dla KEM definiujemy analogicznie do definicji 11.2. Tak jak tam, przeciwnik tutaj podsłuchuje pojedynczy szyfrogram c .

Definicja 11.2 wymaga, aby atakujący nie był w stanie rozróżnić, czy c jest szyfrowaniem jakiejś wiadomości m_0 , czy innej wiadomości m_1 . W przypadku KEM nie ma wiadomości i zamiast tego wymagamy, aby zamknięty klucz k był nie do odróżnienia od jednolitego klucza, który jest niezależny od tekstu zaszyfrowanego c .

Niech $\Pi = (\text{Gen}, \text{Encaps}, \text{Decaps})$ będzie KEM, a A dowolnym przeciwnikiem.

Eksperyment nieroróżnialności CPA KEMcpa $A, \Pi(n)$:

1. Uruchomiono $\text{Gen}(1n)$ w celu uzyskania kluczy (pk, sk) . Następnie uruchamiane jest $\text{Encaps}_{pk}(1n)$ w celu wygenerowania (c, k) z $k \in \{0, 1\}^n$.
2. Wybierany jest bit jednolity $b \in \{0, 1\}$. Jeśli $b = 0$, ustaw $\hat{k} := k$. Jeżeli $b = 1$ to wybierz jednorodny $\hat{k} \in \{0, 1\}^n$.
3. Podaj (pk, c, \hat{k}) A, który wyprowadza bit b . Wynik eksperymentu definiuje się jako 1, jeśli $b = \hat{b}$, i 0 w przeciwnym razie.

W eksperymencie A otrzymuje zaszyfrowany tekst c i albo rzeczywisty klucz k odpowiadający c , albo niezależny, jednolity klucz. KEM jest bezpieczny pod względem CPA, jeśli żaden skuteczny przeciwnik nie jest w stanie rozróżnić tych możliwości.

DEFINICJA 11.11 Mechanizm enkapsulacji klucza Π jest bezpieczny CPA, jeśli dla wszystkich probabilistycznych przeciwników A w czasie wielomianowym istnieje pomijalna funkcja negl taka, że

$$\Pr[\text{KEMcpa}_{A, \Pi}(n) = 1] \leq \frac{1}{2^{n \text{ zaniedbywanie}(n)}}.$$

W dalszej części tej sekcji udowodnimy następujące twierdzenie:

TWIERDZENIE 11.12 Jeśli Π jest KEM zabezpieczonym CPA, a Π jest schematem szyfrowania klucza prywatnego, którego szyfrowanie jest nieroróżnialne w obecności osoby podsłuchującej, to Π_{hy} , jak w Konstrukcji 11.10, jest schematem szyfrowania kluczem publicznym zabezpieczonym CPA.

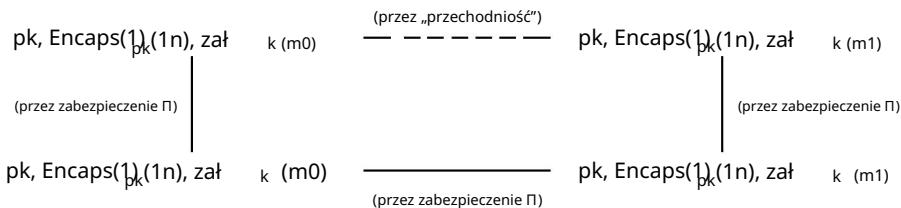
Zanim formalnie udowodnimy twierdzenie, podamy trochę intuicji. Niech zapis „X \xrightarrow{c} Y” oznacza zdarzenie, w którym żaden przeciwnik w czasie wielomianowym nie jest w stanie rozróżnić dwóch rozkładów X i Y. (Koncepcję tę omówiono bardziej formalnie w sekcji 7.8, chociaż nie będziemy się tutaj na niej opierać.)

Na przykład, niech $\text{Encaps}(1)_{pk}(1n)$ (odpowiednio, $\text{Encaps}(2)_{pk}(1n)$) oznacza tekst zaszyfrowany (odpowiednio klucz) wyprowadzany przez Encaps. Oznacza to, że Π jest zabezpieczone przed CPA

$$pk, \text{Encaps}(1)_{pk}(1n), \text{Encaps}(2)_{pk}(1n) \xrightarrow{c} pk, \text{Encaps}(1)_{pk}(1n), k,$$

gdzie pk jest generowane przez $\text{Gen}(1n)$, a k jest wybierane niezależnie i równomiernie z $\{0, 1\}^n$. Podobnie, fakt, że Π ma nieroróżnialne szyfrowanie w obecności podsłuchującego, oznacza, że dla dowolnego m_0, m_1 wyjścia A mamy $\text{Enc}_k(m_0) \xrightarrow{c} \text{Enc}_k(m_1)$, jeśli k jest wybrane jednolicie losowo. Aby udowodnić bezpieczeństwo CPA Π_{hy} , musimy to pokazać

$$pk, \text{Encaps}(1)_{pk}(1n), \text{Zał} \xrightarrow{c} pk, \text{Encaps}(1)_{pk}(1n), \text{Enc}_k(m_1) \quad (11.10)$$



RYSUNEK 11.3: Wysokopoziomowa struktura dowodu Twierdzenia 11.12 (strzałki oznaczają nieroróżnialność).

dla m0, m1 wyjście przez przeciwnika ppt A. (Równanie (11.10) wystarczy, aby pokazać, że Π ma nieroróżnialne szyfrowanie w obecności podsłuchującego, a zgodnie z Twierdzeniem 11.3 oznacza to, że Π jest zabezpieczone CPA.)

Dowód przebiega w trzech etapach. (Patrz rysunek 11.3.) Najpierw to udowodnimy

$$pk, \text{Encaps}(1)_{pk}(1n), \text{Zał } k(m0)(1n), \overset{c}{\text{Encaps}}(1)_{pk} \quad k(m0), (11.11)$$

gdzie po lewej stronie k jest wyprowadzane przez $\text{Encaps}(2)(1n)$, a po prawej k jest niezależnym, jednolitym kluczem. Wynika to z dość prostej redukcji, ponieważ bezpieczeństwo CPA Π oznacza dokładnie, że $\text{Encaps}(2)(1n)$ nie można odróżnić od jednolitego klucza k, nawet biorąc pod uwagę pk i $\text{Encaps}(1)(1n)$.

Następnie to udowodnimy

$$pk, \text{Encaps}(1)_{pk}(1n), \text{Enc } k(m0) \quad \overset{c}{\text{pk, Encaps}(1)_{pk}(1n), \text{Enc } k(m1)} . (11.12)$$

Tutaj różnica polega na szyfrowaniu m0 lub m1 przy użyciu Π i jednolitego, niezależnego klucza k. Zatem równanie (11.12) wynika z faktu, że Π ma nieroróżnialne szyfrowanie w obecności podsłuchującego.

Dokładnie tak jak w przypadku Równania (11.11) również możemy to pokazać

$$pk, \text{Encaps}(1)_{pk} \text{ Enc } k \text{ Encaps}(1)(13), \overset{c}{\text{Enc }} k(m1)(1n)_{pk}$$

ponownie opierając się na bezpieczeństwie CPA Π . Równania (11.11)–(11.13) implikują, poprzez przechodniość, pożądany wynik równania (11.10). (Przechodniość będzie ukryta w dowodzie, który przedstawimy poniżej.)

Przedstawiamy teraz pełny dowód.

DOWÓD (Twierdzenia 11.12) Udowodnimy, że Π ma nieroróżnialne szyfrowanie w obecności osoby podsłuchującej; według Propozycji 11.3 oznacza to, że jest on bezpieczny dla CPA.

Napraw dowolnego przeciwnika ppt Ahy i rozważ eksperyment PubKeav Ahy , Π_{hy} . Naszym celem jest udowodnienie, że istnieje funkcja pomijalna taka , że

$$\Pr[\text{PubKeav Ahy ,}\Pi_{\text{hy}}(n) = 1] = \frac{1}{2}^{\text{zaniechanie}(n)}.$$

Z definicji eksperymentu, mamy

$$\begin{aligned} & \Pr[\text{PubKeav Ahy ,}\Pi_{\text{hy}}(n) = 1] \\ &= \frac{1}{2} \cdot \Pr[A \text{ hy}(pk, \text{Encaps}(1)(pk), \text{Enc } k(m0)) = 0] \\ &\quad + \frac{1}{2} \cdot \Pr[A \text{ hy}(pk, \text{Encaps}(1)(pk), \text{Enc } k(m1)) = 1], \end{aligned} \tag{11.14}$$

gdzie w każdym przypadku k równa się $\text{Encaps}(2)(1n)$. Rozważmy następującego przeciwnika A1 atakującego Π .

Przeciwnik A1:

1. Dane jest A1 (pk, c, \hat{k}).
2. A1 uruchamia Ahy(pk), aby otrzymać dwa komunikaty m0, m1. Następnie A1 oblicza c = Enc k(m0), podaje zaszyfrowany tekst c, c do Ahy i wyprowadza bit b, który wysyła Ahy .

Rozważ zachowanie A1 podczas ataku na Π w eksperymencie KEMcpa A1, $\Pi(n)$.

Gdy w tym eksperymencie b = 0, wówczas dane jest A1 (pk, c, \hat{k}), gdzie c i \hat{k} zostały wyprowadzone przez Encapspk(1n). Oznacza to, że Ahy otrzymuje szyfrogram w postaci c, c = c, Enc k(m0), gdzie k jest kluczem zawartym w c. Więc,

$$\Pr[A1 \text{ wyjścia } 0 | b = 0] = \Pr[A \text{ hy}(pk, \text{Encaps}(1)(1n), \text{Enc } k(m0)) = 0].$$

Z drugiej strony, gdy b = 1 w doświadczeniu KEMcpa A1, $\Pi(n)$, wówczas dane jest A1 (pk, c, \hat{k}) z \hat{k} równomiernym i niezależnym od c. Jeśli oznaczymy taki klucz przez k, oznacza to, że Ahy otrzymuje szyfrogram w postaci c, Enc k(m0) i

$$\Pr[A1 \text{ wyjścia } 1 | b = 1] = \Pr[A \text{ hy}(pk, \text{Encaps}(1)(1n), \text{Enc } k(m0)) = 1].$$

Ponieważ Π jest KEM bezpiecznym dla CPA, istnieje pomijalna funkcja negl1 taka, że

$$\begin{aligned} & \frac{1}{2} + \text{negl1}(n) = \Pr[\text{KEMcpa A1,}\Pi(n) = 1] \\ &= \frac{1}{2} \cdot \Pr[A1 \text{ wyjścia } 0 | b = 0] + \frac{1}{2} \cdot \Pr[A1 \text{ wyjścia } 1 | b = 1] \\ &= \frac{1}{2} \cdot \Pr[A \text{ hy}(pk, \text{Encaps}(1)(pk), \text{Enc } k(m0)) = 0] \\ &\quad + \frac{1}{2} \cdot \Pr[A \text{ hy}(pk, \text{Encaps}(1)(pk), \text{Enc } k(m0)) = 1] \end{aligned} \tag{11.15}$$

gdzie k jest równe $\text{Encaps}(2)(1n)$ a k jest jednolitym i niezależnym kluczem.

Następnie rozważmy następującego przeciwnika ppt A, który podsłuchuje wiadomość zaszyfrowaną przy użyciu schematu klucza prywatnego Π .

Przeciwnik A:

1. A $(1n)$ samodzielnie uruchamia $\text{Gen}(1n)$ w celu wygenerowania kluczy (pk , sk). Oblicza także $c = \text{Encaps}(1)(1n)$.
2. A uruchamia $Ahy(pk)$, aby otrzymać dwa komunikaty m_0, m_1 . Są one wyprowadzane przez A, a w zamian jest przekazywany tekst zaszyfrowany c .
3. A przekazuje szyfrogram c , c do Ahy i wyprowadza bit b , który wysyła Ahy .

Gdy $b = 0$ w eksperymencie $\text{PrivKeav}(n)$, przeciwnik A otrzymuje szyfr- A, Π tekst c , który zaszyfrowaniem m_0 przy użyciu klucza k , który jest jednolity i niezależny od czegokolwiek innego. Zatem Ahy otrzymuje szyfrogram w postaci c , $\text{Enc}_k(m_0)$, gdzie k jest jednolite i niezależne od c , oraz

$$\Pr[A \text{ wyjście } 0 \mid b = 0] = \Pr[A \text{ hy}(pk, \text{Encaps}(1)(1n), \text{Enc}_{pk}^k(m_0)) = 0].$$

Z drugiej strony, gdy $b = 1$ w eksperymencie $\text{PrivKeav}(n)$, wówczas A otrzymuje A , Π zaszyfrowanie m_1 przy użyciu jednolitego, niezależnego klucza k . Oznacza to, że Ahy otrzymuje zaszyfrowany tekst w postaci c , $\text{Enc}_k(m_1)$ i tak

$$\Pr[A \text{ wyjście } 1 \mid b = 1] = \Pr[A \text{ hy}(pk, \text{Encaps}(1)(1n), \text{Enc}_{pk}^k(m_1)) = 1].$$

Ponieważ Π ma nierozróżnialne zaszyfrowanie w obecności osoby podsłuchującej per, istnieje pomijalna funkcja negl taka, że

$$\begin{aligned} \frac{1}{2} + \text{negl}(n) &= \Pr[\text{PrivKeav}(n) \stackrel{\text{def}}{=} 1] \\ &= \frac{1}{2} \cdot \Pr[A \text{ wyjście } 0 \mid b = 0] + \frac{1}{2} \cdot \Pr[A \text{ wyjście } 1 \mid b = 1] \\ &= \frac{1}{2} \cdot \Pr[A \text{ hy}(pk, \text{Encaps}(1)(1n), \text{Enc}_k(m_0)) = 0] \\ &\quad + \frac{1}{2} \cdot \Pr[A \text{ hy}(pk, \text{Encaps}(1)(1n), \text{Enc}_k(m_1)) = 1]. \end{aligned} \tag{11.16}$$

Postępując dokładnie tak, jak dowodziliśmy równanie (11.15), możemy wykazać, że istnieje funkcja pomijalna negl2 taka, że

$$\begin{aligned} \frac{1}{2} + \text{negl2}(n) &= \Pr[\text{KEMcpa } A2, \Pi(n) = 1] \\ &= \frac{1}{2} \cdot \Pr[A2 \text{ wyjście } 0 \mid b = 0] + \frac{1}{2} \cdot \Pr[A2 \text{ wyjście } 1 \mid b = 1] \\ &= \frac{1}{2} \cdot \Pr[A \text{ hy}(pk, \text{Encaps}(1)(1n), \text{Enc}_k(m_1)) = 1] \\ &\quad + \frac{1}{2} \cdot \Pr[A \text{ hy}(pk, \text{Encaps}(1)(1n), \text{Enc}_k(m_1)) = 0]. \end{aligned} \tag{11.17}$$

Sumując równania (11.15)–(11.17) i korzystając z faktu, że suma trzech funkcji pomijalnych jest zaniedbywalna, widzimy, że istnieje funkcja pomijalna taka , że

$$\begin{aligned} & \frac{3}{2} \cdot \Pr[A \text{ hy}(pk, c, \text{zał}) \quad k(m0) = 0] + \Pr[A \text{ hy}(pk, c, Enc) \quad k(m0) = 1] \\ & + \Pr[A \text{ hy}(pk, c, Enc) \quad k(m0) = 0] + \Pr[A \text{ hy}(pk, c, Enc) \quad k(m1) = 1] \\ & + \Pr[A \text{ hy}(pk, c, Enc) \quad k(m1) = 1] + \Pr[A \text{ hy}(pk, c, Enc) \quad k(m1) = 0], \end{aligned}$$

gdzie $c = \text{Encaps}(1)(1n)$ w \prod wszystkich powyższych przypadkach. Zauważ to

$$\Pr[A \text{ hy}(pk, c, Enc) \quad k(m0) = 1] + \Pr[A \text{ hy}(pk, c, Enc) \quad k(m0) = 0] = 1,$$

ponieważ prawdopodobieństwa zdarzeń uzupełniających zawsze sumują się do 1. Podobnie

$$\Pr[A \text{ hy}(pk, c, Enc) \quad k(m1) = 1] + \Pr[A \text{ hy}(pk, c, Enc) \quad k(m1) = 0] = 1.$$

Dlatego,

$$\begin{aligned} & \frac{1}{2} \cdot \Pr[A \text{ hy}(pk, c, \text{zał}) \quad k(m0) = 0] + \Pr[A \text{ hy}(pk, c, Enc) \quad k(m1) = 1] \\ & = \Pr[\text{PubKeav Ahy}, \Pi \text{hy}(n) = 1] \end{aligned}$$

(używając równania (11.14) dla ostatniej równości), udowadniając twierdzenie. ■

11.3.2 Bezpieczeństwo CCA

Jeżeli schemat szyfrowania klucza prywatnego Π sam w sobie nie jest bezpieczny przed atakami z użyciem wybranego tekstu zaszyfrowanego, to (niezależnie od użytego KEM) nie jest również uzyskany hybrydowy schemat szyfrowania Πhy . Jako prosty, ilustrujący przykład założmy, że przyjmujemy Construction 3.17 jako nasz schemat szyfrowania klucza prywatnego. Następnie, pozostawiając KEM nieokreślony, szyfrowanie wiadomości m przez Πhy odbywa się poprzez obliczenie $(c, k) = \text{Encapspk}(1n)$, a następnie wypisanie tekstu zaszyfrowanego

$$c, G(k) \quad m,$$

gdzie G jest generatorem pseudolosowym. Mając zaszyfrowany tekst c , c , atakujący może po prostu odwrócić ostatni bit c , aby uzyskać zmodyfikowany tekst zaszyfrowany, który jest prawidłowym szyfrowaniem m z odwróconym ostatnim bitem.

Naturalnym sposobem rozwiązania tego problemu jest użycie schematu szyfrowania klucza prywatnego zabezpieczonego przez CCA. Ale to wyraźnie nie wystarczy, jeśli KEM jest podatny na ataki wybranym tekstem zaszyfrowanym. Ponieważ nie zdefiniowaliśmy tego pojęcia, uczynimy to teraz.

Podobnie jak w definicji 11.11, wymagamy, aby przeciwnik, któremu podano zaszyfrowany tekst c , nie mógł odróżnić klucza k zawartego w tym zaszyfrowanym tekście od jednolitego i niezależnego klucza k . Teraz jednak dodatkowo umożliwiamy atakującemu zażądanie dekapsulacji wybranych przez siebie szyfrogramów (o ile różnią się one od szyfrogramu wyzwania).

Formalnie niech $\Pi = (\text{Gen}, \text{Encaps}, \text{Decaps})$ będzie KEM o długości klucza n i A przeciwnikiem i rozważ następujący eksperyment:

Eksperyment nieroróżnialności CCA $\text{KEMcca } A, \Pi(n)$:

1. Uruchomiono $\text{Gen}(1n)$ w celu uzyskania kluczy (pk, sk). Następnie uruchamiane jest $\text{Encaps}_{\text{pk}}(1n)$ w celu wygenerowania (c, k) z $k \in \{0, 1\}^n$.
2. Wybierany jest bit jednorodny $b \in \{0, 1\}$. Jeśli $b = 0$, ustaw $\hat{k} := k$. Jeżeli $b = 1$ to wybierz jednorodny $\hat{k} \in \{0, 1\}^n$.
3. A ma dane (pk, c, \hat{k}) i dostęp do wyroczni $\text{Decapssk}(\cdot)$, ale nie może żądać dekapsulacji samego c .
4. A wyprowadza trochę b . Zdefiniowano wynik eksperymentu wynosić 1, jeśli $b = b$, i 0 w przeciwnym razie.

DEFINICJA 11.13 Mechanizm enkapsulacji klucza Π jest bezpieczny w trybie CCA, jeśli dla wszystkich probabilistycznych przeciwników A w czasie wielomianowym istnieje pomijalna funkcja negl taka, że

$$\Pr[\text{KEMcca } A, \Pi(n) = 1] \xrightarrow[2]{\text{zaniebitywanie (n)}} 1$$

Na szczęście możemy wykazać, że użycie KEM zabezpieczonego CCA w połączeniu ze schematem szyfrowania klucza prywatnego zabezpieczonego CCA daje w rezultacie schemat szyfrowania klucza publicznego zabezpieczony przed atakami z użyciem wybranego tekstu zaszyfrowanego.

TWIERDZENIE 11.14 Jeśli Π jest KEM zabezpieczonym CCA i Π jest schematem szyfrowania klucza prywatnego zabezpieczonym CCA, to Π_{hy} , jak w Konstrukcji 11.10, jest schematem szyfrowania kluczem publicznym zabezpieczonym CCA.

Dowód uzyskuje się poprzez odpowiednią modyfikację dowodu Twierdzenia 11.12.

11.4 Szyfrowanie oparte na CDH/DDH

Do tej pory omawialiśmy abstrakcyjnie szyfrowanie kluczem publicznym, ale nie widzieliśmy jeszcze żadnych konkretnych przykładów schematów szyfrowania kluczem publicznym (lub KEM). Tutaj badamy niektóre konstrukcje oparte na problemach Diffiego-Hellmana.

(Problemy Diffiego-Hellmana przedstawiono w rozdziale 8.3.2.)

11.4.1 Szyfrowanie El Gamala

W 1985 roku Taher El Gamal zauważył, że protokół wymiany kluczy Diffiego – Hellmana (por. sekcja 10.3) można dostosować, aby zapewnić schemat szyfrowania klucza publicznego. Przypomnijmy, że w protokole Diffiego-Hellmana Alicja wysyła wiadomość do Boba, a następnie Bob odpowiada wiadomością do Alicji; na podstawie tych wiadomości Alicja i Bob mogą wyprowadzić wspólną wartość k , która jest nie do odróżnienia (dla podsłuchującego) od jednolitego elementu jakiejś grupy G . Możemy sobie wyobrazić Boba używającego tej wspólnej wartości do zaszyfrowania wiadomości m po prostu wysyłając $k \cdot m$ do Alicji; Alicja może wyraźnie odzyskać m , korzystając ze swojej wiedzy o k , a poniżej będziemy argumentować, że podsłuchujący nie dowiaduje się niczego o m .

W schemacie szyfrowania El Gamala po prostu zmieniamy nasze spojrzenie na powyższą interakcję. Początkową wiadomość Alicji postrzegamy jako jej klucz publiczny, a odpowiedź Boba (zarówno jego początkową odpowiedź, jak i $k \cdot m$) jako tekst zaszyfrowany. Bezpieczeństwo CPA oparte na decyzyjnym założeniu Diffiego-Hellmana (DDH) dość łatwo wynika z bezpieczeństwa protokołu wymiany kluczy Diffiego-Hellmana (Twierdzenie 10.3).

W naszym formalnym postępowaniu zaczniemy od stwierdzenia i udowodnienia prostego lematu leżącego u podstaw schematu szyfrowania El Gamala. Niech G będzie grupą skończoną i niech $m \in G$ będzie elementem dowolnym. Lemat stwierdza, że pomnożenie m przez jednorodny element grupy k daje równomiernie rozłożony element grupy k .

Co ważne, rozkład k jest niezależny od m ; oznacza to, że k nie zawiera informacji o m .

LEMMA 11.15 Niech G będzie grupą skończoną i niech $m \in G$ będzie dowolne. Wtedy wybór jednostajnego $k \in G$ i ustawienie $k := k \cdot m$ daje taki sam rozkład dla k jak wybranie jednostajnego $k \in G$. Inaczej mówiąc, dla dowolnego $g \in G$ mamy

$$\Pr[k \cdot m = g] = 1/|G|,$$

gdzie przyjmuje się prawdopodobieństwo jednolitego wyboru $k \in G$.

DOWÓD Niech $g \in G$ będzie dowolne. Następnie

$$\Pr[k \cdot m = g] = \Pr[k = g \cdot m^{-1}].$$

Ponieważ k jest jednorodne, prawdopodobieństwo, że k jest równe stałemu elementowi $g \cdot m^{-1}$, wynosi dokładnie $1/|G|$. ■

Powyższy lemat sugeruje sposób skonstruowania schematu szyfrowania idealnie tajnego klucza prywatnego z przestrzenią wiadomości G . Nadawca i odbiorca współdzielą jako swój tajny klucz jednolity element $k \in G$. Aby zaszyfrować wiadomość $m \in G$, nadawca oblicza zaszyfrowany tekst $k \cdot m$. Odbiorca może odzyskać wiadomość z zaszyfrowanego tekstu k , obliczając $m := k \cdot k^{-1}$. Idealna tajemnica wynika bezpośrednio z powyższego lematu. W rzeczywistości widzieliśmy już ten schemat w innej postaci —schemat jednorazowego szyfrowania padu to

urzeczywistnienie tego podejścia, w którym podstawową grupą jest zbiór ciągów o określonej długości w ramach operacji bitowego XOR.

Możemy dastosować powyższe pomysły do ustalenia klucza publicznego, zapewniając stronom sposób na wygenerowanie wspólnej „losowo wyglądającej” wartości k poprzez interakcję za pośrednictwem kanału publicznego. Powinno to brzmieć znajomo, ponieważ dokładnie to zapewnia protokół Diffiego-Hellmana. Kontynuujemy szczegółowo.

Podobnie jak w podrozdziale 8.3.2, niech G będzie algorytmem działającym w czasie wielomianowym, który przyjmuje jako dane wejściowe $1n$ i (z wyjątkiem możliwie znikomego prawdopodobieństwa) wyprowadza opis grupy cyklicznej G, jej rząd q (przy $q = n$) oraz generator gr. Schemat szyfrowania El Gamal opisano w Construction 11.16.

BUDOWA 11.16

Niech G będzie jak w tekście. Zdefiniuj schemat szyfrowania klucza publicznego w następujący sposób:

- Gen: na wejściu $1n$ uruchom $G(1n)$ aby otrzymać (G, q, g) . Następnie wybierz uniform $x \in \mathbb{Z}_q$ i oblicz $h := g^x$, a kluczem publicznym to G, q, g, h . Klucz prywatny jest G, q, g, x . Przestrzeń wiadomości to G .
- Enc: na wejściu klucz publiczny $pk = G, q, g, h$ i komunikat $m \in G$, wybierz mundur $y \in \mathbb{Z}_q$ i wyprowadź zaszyfrowany tekst

$$y^x g, \quad h^y \cdot m.$$

- Dec: na wejściu klucz prywatny $sk = G, q, g, x$ i szyfrogram $c1, c2$, wyjście

$$m^* := c2/cx^1.$$

Schemat szyfrowania El Gamal.

Aby zobaczyć, że odszyfrowanie się powiodło, niech $c1, c2 = g^y, \quad hy \cdot m$ gdzie $h = g^x$. Następnie

$$m^* = \frac{c2}{(g^y)^x} = \frac{hy \cdot m}{(g^y)^x} = \frac{(gx)y \cdot M}{g^{xy}} = \frac{g^{xy} \cdot M}{g^{xy}} = m.$$

Przykład 11.17 Niech

$q = 83$ i $p = 2q + 1 = 167$, a G oznacza grupę reszt kwadratowych (tzn. kwadratów) modulo p. (Ponieważ p i q są liczbami pierwszymi, G jest podgrupą rzędu q. Zobacz rozdział 8.3.3.)
Z str. Ponieważ rząd G jest liczbą pierwszą, każdy element G z wyjątkiem 1 jest generatorem; weź $g = 22 = 4 \text{ mod } 167$. Założmy, że odbiorca wybiera tajny klucz 37 Z83 i tak klucz publiczny to

$$pk = p, q, g, h = 167, 83, 4, [437 \text{ mod } 167] = 167, 83, 4, 76,$$

gdzie używamy p do reprezentowania G (zakłada się, że odbiorca wie, że grupa jest zbiorem reszt kwadratowych modulo p).

Załóżmy, że nadawca szyfruje wiadomość $m = 65 \pmod{G}$ (notatka $65 = 302 \pmod{167}$ więc 65 jest elementem podgrupy). Jeśli $y = 71$, szyfrogram to

$$[471 \pmod{167}], [7671 \cdot 65 \pmod{167}] = 132, 44.$$

Aby odszyfrować, odbiorca najpierw oblicza $124 = [13237 \pmod{167}]$; wówczas, ponieważ $66 = [124 \cdot 1 \pmod{167}]$, odbiorca odzyskuje $m = 65 = [44 \cdot 66 \pmod{167}]$.

Teraz udowodnimy bezpieczeństwo schematu. (Czytelnik może chcieć porównać dowód poniższego do dowodów twierdzeń 3.18 i 10.3.)

TWIERDZENIE 11.18 Jeśli problem DDH jest trudny w stosunku do G , wówczas schemat szyfrowania El Gamala jest bezpieczny CPA.

DOWÓD Niech Π oznacza schemat szyfrowania El Gamala. Udowodnimy, że Π ma nieroróżnialne szyfrowanie w obecności podsłuchującego; według Propozycji 11.3 oznacza to, że jest on bezpieczny dla CPA.

Niech A będzie probabilistycznym przeciwnikiem wielomianowym w czasie. Chcemy pokazać, że istnieje funkcja pomijalna taka, że

$$\Pr_{\substack{\text{PubKeav } A, \Pi(n) = 1}} \left[\frac{1}{2} \right] \text{ zaniedbywanie}(n).$$

Rozważmy zmodyfikowany „schemat szyfrowania” Π , gdzie Gen jest taki sam jak w Π , ale szyfrowanie wiadomości m w odniesieniu do klucza publicznego G, q, g, h odbywa się poprzez wybranie jednolitego $y, z \in Z_q$ i wypisanie tekstu zaszyfrowanego

$$y^g, gz \cdot m.$$

Chociaż Π w rzeczywistości nie jest schematem szyfrowania (ponieważ odbiorca nie ma możliwości odszyfrowania), eksperyment $\text{PubKeav}_{A, \Pi}(n)$ jest nadal dobrze zdefiniowany, ponieważ eksperyment ten zależy wyłącznie od algorytmów generowania klucza i szyfrowania.

Z lematu 11.15 i dyskusji, która następuje bezpośrednio po nim wynika, że drugi składnik szyfrogramu w schemacie Π jest elementem grupy o równomiernym rozkładzie i w szczególności jest niezależny od szyfrowanej wiadomości m . (Pamiętaj, że g jest jednolitym elementem G , gdy z jest wybrane jednolicie z Z_q .) Pierwszy składnik szyfrogramu jest trywialnie niezależny od m . Podsumowując, oznacza to, że cały zaszyfrowany tekst nie zawiera żadnych informacji o m . Wynika, że

$$\Pr_{\substack{\text{PubKeav } A, \Pi(n) = 1}} [1] = \frac{1}{2}.$$

Rozważmy teraz następujący algorytm ppt D , który próbuje rozwiązać problem DDH względem G . Przypomnijmy, że D otrzymuje (G, q, g, h_1, h_2, h_3) gdzie $h_1 = g \cdot h_2 = g \cdot z$ (dla jednorodnych x, y, z); ~~o której D jest dostosowanie, który przypadek ma miejsce.~~

Algorytm D: Jako dane

wejściowe podaje się algorytm $(G, q, g, h1, h2, h3)$.

- Ustaw $pk = G, q, g, h1$ i uruchom $A(pk)$, aby otrzymać dwie wiadomości $m0, m1$ G. • Wybierz jednolity bit b i
- ustaw $c1 := h2 \cdot mb$ • Podaj zaszyfrowany tekst $c1, c2$ A i uzyskaj bit wyjściowy b .

Jeśli $b = b$, wyjście 1; w przeciwnym razie wyjście 0.

Przeanalizujmy zachowanie D. Istnieją dwa przypadki do rozważenia: Przypadek 1:

Założmy, że dane wejściowe do D są generowane poprzez uruchomienie $G(1n)$ w celu uzyskania (G, q, g) , a następnie wybranie jednorodnych $x, y, z \in \mathbb{Z}_q$ i na koniec ustawienie $h1 := g^x, h2 := g^y, h3 := g^z$. Następnie D uruchamia A na kluczu publicznym skonstruowanym jako

$$pk = G, q, g, gx$$

oraz tekst zaszyfrowany skonstruowany jako

$$c1, c2 = sol \quad , \quad gy, gz \cdot mb.$$

Widzimy, że w tym przypadku widok A, uruchomiony jako podprogram przez D, rozkłada się identycznie jak widok A w eksperymencie PubKeav(n). Ponieważ D wyprowadza 1 A, Π dokładnie wtedy, gdy wynik b A jest równy b, mamy to

$$\Pr[D(G, q, g, gx, gy, gz) = 1] = \Pr[\text{PubKeav}_{A, \Pi}(n) = 1] = 2^{-1}.$$

Przypadek 2: Założmy, że dane wejściowe do D są generowane poprzez uruchomienie $G(1n)$ w celu uzyskania (G, q, g) , następnie wybranie jednakowych $x, y \in \mathbb{Z}_q$ i na koniec ustawienie $h1 := g^x, h2 := g^y, h3 := g^{xy}$. Następnie D uruchamia A na kluczu publicznym skonstruowanym jako

$$pk = G, q, g, gx$$

oraz tekst zaszyfrowany skonstruowany jako

$$c1, c2 = sol \quad , \quad gxy \cdot mb = g^y, (g^x)y \cdot mb.$$

Widzimy, że w tym przypadku widok A, gdy jest on wykonywany jako podprogram przez D, rozkłada się identycznie jak widok A w eksperymencie PubKeav A, Π(n). Ponieważ D wyprowadza 1 dokładnie wtedy, gdy wynik b A jest równy b, mamy to

$$\Pr[D(G, q, g, gx, gy, gxy) = 1] = \Pr[\text{PubKeav}_{A, \Pi}(n) = 1].$$

Przy założeniu, że problem DDH jest trudny w porównaniu z G, istnieje jest funkcją pomijalną negl taką, że

$$\begin{aligned} negl(n) &= \Pr[D(G, q, g, gx, gy, gz) = 1] = \Pr[D(G, q, g, gx, gy, gxy) = 1] \\ &= \frac{1}{2} \Pr[\text{PubKeav}_{A, \Pi}(n) = 1]. \end{aligned}$$

Oznacza to $\Pr[\text{PubKeav}_{A, \Pi}(n) = 1] = negl(n)$, kończąc dowód. ■

Problemy z wdrażaniem El Gamal

Pokrótce omawiamy kilka praktycznych zagadnień związanych z szyfrowaniem El Gamal.

Udostępnianie parametrów publicznych. Nasz opis schematu szyfrowania El Gamala w Construction 11.16 wymaga, aby odbiornik uruchomił G w celu wygenerowania G, q, g . W praktyce często zdarza się, że parametry te są generowane i ustalane „raz na zawsze”, a następnie udostępniane wielu odbiorcom. (Oczywiście każdy odbiorca x) musi wybrać własną wartość tajną x i opublikować swój własny klucz publiczny $h = g^x$. Na przykład NIST opublikował zestaw zalecanych parametrów odpowiednich do zastosowania w schemacie szyfrowania El Gamala. Udostępnianie parametrów w ten sposób nie wpływa na bezpieczeństwo (zakładając, że parametry zostały wygenerowane poprawnie i uczciwie). Patrząc w przyszłość, zauważamy, że różni się to od przypadku RSA, w którym nie można bezpiecznie udostępniać parametrów (patrz sekcja 11.5.1).

Wybór grupy. Jak omówiono w podrozdziale 8.3.2, rząd grupowy q jest zazwyczaj wybierany jako rząd pierwszy. W przypadku określonych grup coraz popularniejszym wyborem są krzywe eliptyczne; alternatywą jest pozwolenie G na podgrupę Z dla p pierwszego. W sekcji 9.3 znajduje się tabela zalecanych długości kluczy umożliwiających osiągnięcie różnych poziomów bezpieczeństwa.

Przestrzeń wiadomości. Niewygodnym aspektem schematu szyfrowania El Gamala jest to, że przestrzeń wiadomości stanowi grupę G , a nie ciągi bitów o określonej długości. W przypadku niektórych opcji grupy można rozwiązać ten problem, definiując odwracalne kodowanie ciągów bitów jako elementów grupy. W takich przypadkach nadawca może najpierw zakodować swoją wiadomość $m \in \{0, 1\}$ jako element grupy $\hat{m} \in G$, a następnie zastosować szyfrowanie El Gamal do \hat{m} . Odbiorca może odszyfrować, jak w Konstrukcji 11.16, aby otrzymać zakodowaną wiadomość \hat{m} , a następnie odwrócić kodowanie, aby odzyskać oryginalną wiadomość m .

Prostszym podejściem jest użycie (wariantu) szyfrowania El Gamal jako części hybrydowego schematu szyfrowania. Na przykład nadawca może wybrać jednolity element grupy $m \in G$, zaszyfrować go przy użyciu schematu szyfrowania El Gamala, a następnie zaszyfrować swoją właściwą wiadomość przy użyciu schematu szyfrowania klucza prywatnego i klucza $H(m)$, gdzie $H : G \times \{0, 1\}^n \rightarrow \{0, 1\}^m$ jest odpowiednią funkcją wyprowadzania klucza (patrz następna sekcja). W tym przypadku bardziej efektywne byłoby użycie KEM opartego na DDH, który opiszemy poniżej.

11.4.2 Enkapsulacja kluczy oparta na DDH

Na końcu poprzedniej sekcji zauważaliśmy, że szyfrowanie El Gamala można wykorzystać jako część schematu szyfrowania hybrydowego, po prostu szyfrując jednolity element grupy m i używając skrótu tego elementu jako klucza. Ale to jest marnotrawstwo! Dowód bezpieczeństwa szyfrowania El Gamal pokazuje, że c_1

(gdzie c_1 jest pierwszym składnikiem szyfrogramu, a x jest kluczem prywatnym odbiorcy) jest już nie do odróżnienia od jednolitego elementu grupy, więc nadawca/odbiorca również dobrze może go użyć. Konstrukcja 11.19 ilustruje KEM

podąża za tym podejściem. Należy zauważać, że wynikowa enkapsulacja składa się tylko z jednego elementu grupy. Natomiast gdybyśmy zastosowali szyfrowanie El Gamala dla jednolitego elementu grupowego, tekst zaszyfrowany zawierałby dwa elementy grupowe.

BUDOWA 11.19

Niech G będzie takie jak w poprzedniej sekcji. Zdefiniuj KEM w następujący sposób:

- Gen: na wejściu $1n$ uruchom $G(1n)$ aby otrzymać (G, q, g) . Wybrać mundur (n)
 - x Z q i ustaw $h := g^x$. Określ także funkcję $H : G \rightarrow \{0, 1\}$
 - jakiejś funkcji (patrz tekst). Klucz publiczny to G, q, g, h , a klucz prywatny to G, q, g ,
 - x. Encaps: po wprowadzeniu klucza
- publicznego $pk = G, q, g, h, H$ wybierz uniform
 - y Z q i wypisz zaszyfrowany tekst y i klucz $H(h|y)$.
- Decaps: po wprowadzeniu klucza prywatnego $sk = G, q, g, x$ i tekstu zaszyfrowanego c
 G , wyprowadź klucz $H(c|x)$.

KEM „podobny do El Gamala”.

Jak opisano, konstrukcja pozostawia funkcję wyprowadzania klucza H nieokreślona i istnieje kilka opcji tej funkcji. (Więcej informacji na temat wyprowadzania klucza znajdziesz w sekcji 5.6.4.) Jedną z możliwości jest wybranie funkcji $H : G \rightarrow \{0, 1\}$, która jest (blisko) regularna, co oznacza, że dla każdego możliwego klucza $k \in \{0, 1\}$ liczba elementów grupy przypisanych do k jest w przybliżeniu taka sama. (Formalnie potrzebujemy pomijalnej funkcji negl takiej, że dla każdego $k \in \{0, 1\}$

$$2 \cdot \Pr[H(g) = k] \geq \text{negl}(n),$$

gdzie przyjmuje się prawdopodobieństwo równomiernego wyboru $g \in G$. Gwarantuje to, że rozkład na kluczu k jest statystycznie bliski równomierнемu.) Zarówno złożoność H , jak i osiągalna długość klucza będą zależeć od konkretnej grupy G używanej.

Druga możliwość polega na tym, aby H było funkcją kluczową, gdzie (jednolity) klucz dla H jest zawarty jako część klucza publicznego odbiorcy. Działa to, jeśli H jest silnym ekstraktorem, jak wspomiano krótko w sekcji 5.6.4. Właściwy wybór tutaj (aby mieć pewność, że otrzymany klucz będzie statystycznie zbliżony do jednolitego) będzie zależał od wielkości G .

W każdym z powyższych przypadków dowód bezpieczeństwa CPA oparty na decyzyjnym założeniu Diffiego-Hellmana (DDH) jest łatwy do zrealizowania poprzez dostosowanie dowodu bezpieczeństwa dla protokołu wymiany kluczy Diffiego-Hellmana (Twierdzenie 10.3).

TWIERDZENIE 11.20 Jeśli problem DDH jest trudny w stosunku do G i H zostanie wybrane zgodnie z opisem, wówczas Konstrukcja 11.19 jest KEM bezpiecznym dla CPA.

Jeśli ktoś chce modelować H jako losową wyrocznię, wówczas można udowodnić, że konstrukcja 11.19 jest bezpieczna pod względem CPA w oparciu o (słabsze) obliczeniowe założenie Diffiego-Hellmana (CDH). Omawiamy to w poniższej sekcji.

11.4.3 *KEM oparty na CDH w modelu Random-Oracle

W tej sekcji pokazujemy, że jeśli ktoś chce modelować H jako losową wyrocznię, wówczas można udowodnić, że konstrukcja 11.19 jest bezpieczna pod względem CPA w oparciu o założenie CDH.

(Czytelnicy mogą przeczytać sekcję 5.5, aby przypomnieć sobie model losowej wyroczni). Intuicyjnie założenie CDH implikuje, że x (z klucza publicznego) i tekst zaszyfrowany $c = g$ atakujący obserwując $h =$ atakująca nie może zadać pytania g nie może obliczyć $DHg(h, c) = \text{godz } y$. W szczególności osoba y losowej wyroczni. Oznacza to jednak, że zamknięty klucz $H(y)$ jest całkowicie losowy z punktu widzenia atakującego. Intuicja ta została przekształcona poniżej w formalny dowód.

Jak wskazuje powyższa intuicja, dowód zasadniczo opiera się na modelowaniu H jako losowej wyroczni³. W szczególności dowód opiera się na faktach, że (1) jedynym sposobem poznania $H(y)$ jest jawne zapytanie h y do H , co oznaczałoby, że atakujący rozwiązał instancję CDH (w sekcji 5.5.1 nazywa się to „możliwością wyodrębnienia”) oraz (2) jeśli atakujący nie wysyła zapytania h y do H, wówczas wartość $H(y)$ wynosi jednolite z punktu widzenia atakującego. Te właściwości obowiązują tylko – w istocie mają sens tylko – jeśli H jest modelowany jako losowa wyrocznia.

TWIERDZENIE 11.21 Jeśli problem CDH jest trudny w stosunku do G, a H jest modelowany jako losowa wyrocznia, wówczas Konstrukcja 11.19 jest zabezpieczona CPA.

DOWÓD Niech Π oznacza Konstrukcję 11.19 i niech A będzie przeciwnikiem ppt .

Chcemy pokazać, że istnieje funkcja pomijalna taka , że

$$\Pr[\text{KEMcpa } A, \Pi(n) = 1] \xrightarrow[+]{\text{zaniechanie}(n), 2}$$

Powyższe prawdopodobieństwo uwzględnia także jednolity wybór funkcji H, do której A ma dostęp wyroczni.

Rozważmy wykonanie eksperymentu KEMcpa $A, \Pi(n)$ w którym klucz publiczny i niech to G, q, g, h , a szyfrogram to $c = g$ zapytania $DHg(h, c) = y$, Query będzie zdarzeniem, które $A = h$ y do H. Mamy

$$\begin{aligned} \Pr[\text{KEMcpa } A, \Pi(n) = 1] &= \Pr[\text{KEMcpa } A, \Pi(n) = 1 \mid \text{Zapytanie}] \\ &\quad + \Pr[\text{KEMcpa } A, \Pi(n) = 1 \mid \text{Zapytanie}] \\ &= \Pr[\text{KEMcpa } A, \Pi(n) = 1 \mid \text{Zapytanie}] + \Pr[\text{Zapytanie}]. \quad (11.18) \end{aligned}$$

Jeśli $\Pr[\text{Query}] = 0$, to $\Pr[\text{KEMcpa } A, \Pi(n) = 1 \mid \text{Zapytanie}] = 0$. W przeciwnym razie

$$\Pr[\text{KEMcpa } A, \Pi(n) = 1 \mid \text{Zapytanie}] = \Pr[\text{KEMcpa } A, \Pi(n) = 1 \mid \text{Zapytanie}] \cdot \Pr[\text{Zapytanie}]$$

$$= \Pr[\text{KEMcpa } A, \Pi(n) = 1 \mid \text{Zapytanie}]$$

³jest to prawdą, o ile chcemy opierać się wyłącznie na założeniu CDH. Jak zauważono wcześniej, dowód bez przypadkowych wyroczni jest możliwy, jeśli opieramy się na silniejszym założeniu DDH.

W eksperymencie $\text{KEMcpa}_{\Pi}(n)$, przeciwnik A otrzymuje klucz publiczny i szyfrogram plus albo zakodowany klucz k Query nie $\overset{\text{def}}{=} H(h \cdot y)$ lub klucz jednolity. Jeżeli występuje, wówczas k jest równomiernie rozłożone z perspektywy przeciwnika, a zatem nie ma możliwości, aby A mógł rozróżnić te dwie możliwości. To znaczy że

$$\Pr[\text{KEMcpa}_{\Pi}(n) = 1 \mid \text{Zapytanie}] = \frac{1}{2}.$$

Wracając do równania (11.18), mamy zatem

$$\Pr[\text{KEMcpa } A, \Pi(n) = 1] = \frac{1}{2} + \Pr[\text{Zapytanie}].$$

Następnie pokażemy, że $\Pr[\text{Query}]$ jest nieistotne, co kończy dowód.

Niech $t = t(n)$ będzie (wielomianem) górną granicą liczby zapytań, które A wykonuje do losowej wyroczni H. Zdefiniuj następujący algorytm ppt A dla problemu CDH względem G:

Algorytm A: Na wejściu algorytmu podano G, q, g, h, c.

- Ustaw $pk = G, q, g, h$ i wybierz jednorodność k $\in \{0, 1\}$.
- Uruchom $A(pk, c, k)$. Kiedy A wysyła zapytanie do H, odpowiedz na nie, wybierając świeży, jednolity ciąg znaków.
- Na końcu wykonywania A niech y_1, \dots, y_t to będzie lista zapytań, które A skierował do H. Wybierz jednolity indeks i $\in \{1, \dots, t\}$ i wypisz y_i .

Interesuje nas prawdopodobieństwo, z jakim A rozwiąże problem CDH, czyli $\Pr[A(G, q, g, h, c) = DHg(h, c)]$, gdzie prawdopodobieństwo przejmuje G, q, g wynik przez G(1n), jednorodność h, c $\in G$ i losowość A. Aby przeanalizować to prawdopodobieństwo, zauważmy najpierw, że zapytanie o zdarzenie jest nadal dobrze zdefiniowane podczas wykonywania A, mimo że A nie może wykryć, czy ono występuje. Co więcej, prawdopodobieństwo zdarzenia Query , gdy A jest uruchamiane jako podprogram przez A, jest identyczne z prawdopodobieństwem zdarzenia Query w eksperymencie $\text{KEMcpa}(n)$. Wynika to z faktu, że widok A jest identyczny obu przypadkach, aż do wystąpienia zdarzenia Query : w każdym przypadku G, q, g są wyprowadzane przez G(1n); w każdym przypadku h i c są jednolitymi elementami G, a k jest jednolitym ciągiem -bitowym; w każdym przypadku na zapytania do H inne niż $H(DHg(h, c))$ odpowiada się jednolitym ciągiem bitów. (W $\text{KEMcpa}_{\Pi}(n)$ na zapytanie $H(DHg(h, c))$ odpowiada się faktycznie zamkniętym kluczem, który jest równy k z prawdopodobieństwem 1/2, natomiast gdy A jest uruchamiane jako podprogram przez A na zapytanie $H(DHg(h, c))$ odpowiada jednolity ciąg -bitowy, który jest niezależny od k. Jednak po wykonaniu tego zapytania następuje zdarzenie Query .)

Na koniec zauważ, że gdy pojawią się Query , to $DHg(h, c) \in \{y_1, \dots, y_t\}$ z definicji, więc A wyprowadza poprawny wynik $DHg(h, c)$ z prawdopodobieństwem równym

przynajmniej $1/t$. Dlatego stwierdzamy, że

$$\Pr[A(G, q, g, h, c) = \text{DHg}(h, c)] = \Pr[\text{Zapytanie}]/t,$$

lub $\Pr[\text{Zapytanie}] = t \cdot \Pr[A(G, q, g, h, c) = \text{DHg}(h, c)] = \Pr[\text{Zapytanie}]$. Ponieważ problem CDH jest trudny dla G , to drugie prawdopodobieństwo jest znikome; ponieważ t jest wielomianem, oznacza to, że $\Pr[\text{Query}]$ jest również nieistotne. To kończy dowód. ■

W następnej sekcji zobaczymy, że można wykazać, że Konstrukcja 11.19 jest bezpieczna pod względem CCA przy silniejszym wariancie założenia CDH (jeśli będziemy nadal modelować H jako losową wyrocznię).

11.4.4 Bezpieczeństwo wybranego tekstu zaszyfrowanego i DHIES/ECIES

Schemat szyfrowania El Gamal jest podatny na ataki z wykorzystaniem wybranego tekstu zaszyfrowanego. Wynika to z faktu, że jest plastyczny. Przypomnijmy, że schemat szyfrowania jest plastyczny, nieformalnie, jeśli otrzymamy zaszyfrowany tekst c , który jest szyfrowaniem jakiejś nieznanej wiadomości m , możliwe jest wygenerowanie zmodyfikowanego tekstu zaszyfrowanego c' , który jest zaszyfrowaniem wiadomości m mającej pewien znany związek z m .

W przypadku szyfrowania El Gamala rozważmy przeciwnika A , który przechwytuje zaszyfrowany tekst $c = c_1, c_2$ zaszyfrowany kluczem publicznym $pk = G, q, g, h$, a następnie konstruuje zmodyfikowany tekst zaszyfrowany $c' = c_1, c_2$ dla niektóre $a \in G$. Jeśli c jest szyfrowaniem wiadomości $m \in G$ (co może być nieznane A), mamy $c_1 = g \cdot m$ dla pewnego $y \in \mathbb{Z}_q$. Ale wtedy $y \in c_2 = h \cdot y$

$$c_1 = r \cdot y \text{ i } c_2 = h \cdot y \cdot (a \cdot m),$$

i tak c' jest prawidłowym szyfrowaniem wiadomości $a \cdot m$. Innymi słowy, A może przekształcić szyfrowanie (nieznanej) wiadomości m w szyfrowanie (nieznanej) wiadomości $a \cdot m$. Jak omówiono w Scenariuszu 3 w Sekcji 11.2.3, tego rodzaju atak może mieć poważne konsekwencje.

KEM omówiony w poprzedniej sekcji może być również plastyczny w zależności od konkretnej używanej funkcji wyprowadzania klucza H . Jeśli jednak H zamodeluje się jako losową wyrocznię, wówczas takie ataki nie wydają się już możliwe. W rzeczywistości można w tym przypadku udowodnić, że konstrukcja 11.19 jest zabezpieczona CCA w oparciu o tak zwane założenie CDH luki. Przypomnijmy, że założenie CDH mówi y (dla pewnego generatora x i g obliczyły g^y). Założenie CDH o luce g , nie jest możliwe, aby dane elementy grupy g mówią, że pozostaje to niewykonalne nawet przy dostępie do wyroczni O w taki sposób, że $O(U, V)$ zwraca 1 dokładnie wtedy, gdy $V = U$ y Inaczej mówiąc, problem CDH pozostaje trudny nawet biorąc pod uwagę wyrocznię, która rozwiązuje problem DDH. (Nie podajemy formalnej definicji, ponieważ nie będziemy używać tego założenia w dalszej części książki.) Uważa się, że to założenie obowiązuje w przypadku klas grup omawianych w tej książce. Dowód poniższego jest bardzo podobny do dowodu Twierdzenia 11.38.

TWIERDZENIE 11.22 Jeśli problem CDH z przerwami jest trudny w porównaniu z G, a H jest modelowane jako losowa wyrocznia, to Konstrukcja 11.19 jest KEM zabezpieczonym przez CCA.

Warto zauważyć, że tę samą konstrukcję (mianowicie Konstrukcja 11.19) można analizować przy różnych założeniach i w różnych modelach, uzyskując różne wyniki. Zakładając jedynie, że problem DDH jest trudny (i dla H odpowiednio wybrany), schemat jest bezpieczny dla CPA. Jeśli zamodelujemy H jako losową wyrocznię (która nakłada na H bardziej rygorystyczne wymagania), to przy słabszym założeniu CDH otrzymamy bezpieczeństwo CPA, a przy silniejszym założeniu CDH luki otrzymamy bezpieczeństwo CCA.

BUDOWA 11.23

Niech G będzie jak w tekście. Niech $\Pi_E = (\text{Enc}, \text{Dec})$ będzie schematem szyfrowania klucza prywatnego i niech $\Pi_M = (\text{Mac}, \text{Vrfy})$ będzie kodem uwierzytelniającym wiadomość. Zdefiniuj schemat szyfrowania klucza publicznego w następujący sposób:

- Gen: Na wejściu $1n$ uruchom $G(1n)$, aby otrzymać (G, q, g) . wybierz mundur. The $x \in \mathbb{Z}_q$, ustaw $h := g^x$ i podaj funkcję $H : G \rightarrow \{0, 1\}$ klucz publiczny to G, q, g, h, H , 2^n a klucz prywatny to G, q, g, x, h .
- Enc: Po wprowadzeniu klucza publicznego $pk = G, q, g, h, H$, wybierz uniform (m) i $y \in \mathbb{Z}_q$ i ustaw $kEM := H(h^y)$. Oblicz $c = \text{Enc } kE$ wyrowadź zaszyfrowany tekst $g^{y \cdot \text{Załącznik } kE}$, $C = \text{MackM}(c)$.

Dec: Na wejściu klucz prywatny $sk = G, q, g, x, h$ i tekst zaszyfrowany, t , wyjście jeśli $c = G$.
 $c, C \in \mathbb{Z}_q$ W innym przypadku oblicz $kEM := H(c \cdot x)$. Jeśli
 $\text{VrfyM}(c, t) = 1$, następnie wyrowadź m ; w przeciwnym razie wyrowadź $\text{Dec } kE(c)$.

DHIES/ECIES.

Szyfrowanie bezpieczne CCA w Construction 11.19. Połączenie KEM w Construction 11.19 z dowolnym schematem szyfrowania klucza prywatnego zabezpieczonego przez CCA daje schemat szyfrowania klucza publicznego zabezpieczonego przez CCA. (Patrz Twierdzenie 11.14.) Utworzenie tego podejścia przy użyciu konstrukcji 4.18 dla komponentu klucza prywatnego odpowiada temu, co dzieje się w DHIES/ECIES, którego warianty są zawarte w standardzie ISO/IEC 18033-2 dotyczącym szyfrowania klucza publicznego. (Patrz Konstrukcja 11.23.) Szyfrowanie wiadomości m w tych schematach ma postać

$$y \in \mathbb{Z}_q, \text{ Załącznik } kE(m), \text{ MackM}(c),$$

gdzie Enc oznacza schemat szyfrowania klucza prywatnego z zabezpieczeniem CPA, a c oznacza $\text{Enc } kE(M)$. DHIES, zintegrowany schemat szyfrowania Diffiego – Hellmana, może być używany ogólnie w odniesieniu do dowolnego schematu tej postaci lub w szczególności w odniesieniu do przypadku, gdy grupa G jest cykliczną podgrupą pola skończonego. ECIES, zintegrowany schemat szyfrowania krzywej eliptycznej, odnosi się do przypadku, gdy G jest grupą o krzywej eliptycznej. Zauważamy, że w konstrukcji 11.23 niezwykle ważne jest sprawdzenie podczas deszyfrowania, czy c , pierwszy składnik tekstu zaszyfrowanego, znajduje się w G . W przeciwnym razie osoba atakująca może zażądać odszyfrowania zniekształconego

szłyfrogram $c, c \rightarrow G$, t w którym $c \in G$; odszyfrowanie takiego zaszyfrowanego tekstu (tj. bez zwieraczący) może spowodować wyciek informacji o kluczu prywatnym.

Zgodnie z Twierdzeniem 4.19, zaszyfrowanie wiadomości, a następnie zastosowanie (silnego) kodu uwierzytelniającego wiadomość daje schemat szylfowania kluczem prywatnym zabezpieczonym przez CCA. Łącząc to z Twierdzeniem 11.14, dochodzimy do wniosku:

DODATEK 11.24 Niech ΠE będzie schematem szylfowania klucza prywatnego z zabezpieczeniem CPA i niech ΠM będzie silnie zabezpieczonym kodem uwierzytelniającym wiadomość. Jeśli problem CDH z przerwami jest trudny w stosunku do G , a H jest modelowany jako losowa wyrocznia, wówczas Konstrukcja 11.23 jest schematem szylfowania klucza publicznego zabezpieczonym przez CCA.

11.5 Szyfrowanie RSA

W tej sekcji zwracamy uwagę na schematy szylfowania oparte na założeniu RSA zdefiniowanym w sekcji 8.2.4. Zauważamy, że chociaż szylfowanie oparte na RSA jest obecnie w powszechnym użyciu, obecnie następuje stopniowe odchodzenie od stosowania RSA – w kierunku korzystania z systemów kryptograficznych opartych na CDH/DDH, opierających się na grupach krzywych eliptycznych – ze względu na dłuższe długości kluczów wymagane w przypadku schematów opartych na RSA. W celu dalszej dyskusji odsyłamy do Sekcji 9.3.

11.5.1 Zwykły RSA

Zaczynamy od opisu prostego schematu szylfowania opartego na problemie RSA. Choć schemat jest niepewny, stanowi użyteczny punkt wyjścia dla kolejnych bezpiecznych schematów.

Niech GenRSA będzie algorytmem ppt , który na wejściu $1n$ generuje moduł N będący iloczynem dwóch n-bitowych liczb pierwszych wraz z liczbami całkowitymi e , d spełniającymi $ed = 1 \bmod \varphi(N)$. (Jaki zwykły algorytm może zawieść z znakomitym prawdopodobieństwem, ale tutaj to ignorujemy.) Przypomnijmy z podrozdziału 8.2.4, że taki algorytm można łatwo skonstruować z dowolnego algorytmu GenModulus , który generuje moduł złożony N wraz z jego rozkładem na czynniki; patrz Algorytm 11.25.

ALGORYTM 11.25 Generowanie

klucza RSA GenRSA

Wejście: Parametr bezpieczeństwa $1n$

Wyjście: N, e, d zgodnie z opisem w tekście

$(N, p, q) \leftarrow \text{GenModulus}(1n)$

$\varphi(N) := (p - 1)(q - 1)$

wybierz $e > 1$ takie, że $\gcd(e, \varphi(N)) = 1$ oblicz $d :=$

$[e^{-1} \bmod \varphi(N)]$ powrót N, e, d

Niech N, e, d będzie jak powyżej i niech $c = me \bmod N$. Szyfrowanie RSA opiera się na fakcie, że ktoś, kto zna d , może odzyskać m z c za pomocą obliczeń [$c^d \bmod N$]; to działa, ponieważ

$$c^d = (ja)^d \bmod N = med = m \bmod N,$$

jak omówiono w sekcji 8.2.4. Z drugiej strony, bez znajomości d (nawet jeśli znane są N i e), założenie RSA (por. Definicja 8.46) implikuje, że trudno jest odzyskać m z c , przynajmniej jeśli m jest wybrane jednolicie z Z

N . To naturalnie sugeruje schemat szyfrowania kluczem publicznym pokazany jako Konstrukcja 11.26: Odbiornik uruchamia GenRSA, aby uzyskać N, e, d ; publikuje N i e jako swój klucz publiczny, a d przechowuje w swoim kluczu prywatnym. Aby zaszyfrować wiadomość m – Z nadawca oblicza tekst zaszyfrowany $c := [me \bmod N]$. Ponieważ mamy właśnie N , odbiorca – kto wie d – może odszyfrować c i odzyskać m .

BUDOWA 11.26

Niech GenRSA będzie jak w tekście. Zdefiniuj schemat szyfrowania klucza publicznego w następujący sposób:

- Gen: na wejściu $1n$ uruchom GenRSA($1n$), aby otrzymać N, e i d . Klucz publiczny to N, e , a klucz prywatny to N, d .
- Enc: na wejściu klucz publiczny $pk = N, e$ i komunikat $m \in Z^{N}$, oblicz szyfrogram $c := [m^e \bmod N]$.
- Dec: na wejściu klucz prywatny $sk = N, d$ i szyfrogram $c \in Z^{N}$, obliczyć wiadomość $m := [ok^d \bmod N]$.

Zwykły schemat szyfrowania RSA.

Poniżej przedstawiono praktyczny przykład powyższego (patrz także Przykład 8.49).

Przykład 11.27

Załóżmy, że wyjścia GenRSA (N, e, d) = (391, 3, 235). (Zauważ, że $391 = 17 \cdot 23$ i tak $\varphi(391) = 16 \cdot 22 = 352$. Co więcej, $3 \cdot 235 = 1 \bmod 352$.) Zatem klucz publiczny to (391, 3), a klucz prywatny to (391, 235).

Aby zaszyfrować wiadomość $m = 158 \in Z_{391}$ używając klucza publicznego (391, 3), po prostu obliczamy $c := [158^3 \bmod 391] = 295$; to jest szyfrogram. Aby odszyfrować, odbiorca oblicza $[295^{235} \bmod 391] = 158$.

Czy zwykły schemat szyfrowania RSA jest bezpieczny? Założenie faktoringu oznacza, że jest to niewykonalne obliczeniowo dla atakującego, który ma

⁴Zakładamy, że $m \in Z_m \subset Z_N$. Jeśli rozkład na czynniki N jest trudny, obliczeniowo trudno jest znaleźć $\{1, \dots, N-1\}$ gdzie $m \in Z_N$ (ponieważ $\gcd(m, N)$ jest nietrywialnym czynnikiem N).

klucz publiczny do uzyskania odpowiedniego klucza prywatnego; patrz Sekcja 8.2.5. Jest to konieczne —ale niewystarczające —aby schemat szyfrowania klucza publicznego był bezpieczny. Założenie RSA implikuje, że jeśli wiadomość m zostanie wybrana jednolicie, to

N osoba podsłuchująca mając podane N, e i c (mianowicie klucz publiczny, a od Z zaszyfrowany tekst) nie będzie w stanie odzyskać m. Są to jednak słabe gwarancje i znacznie odbiegają od poziomu bezpieczeństwa, jakiego oczekujemy! W szczególności pozostawiają otwartą możliwość odzyskania wiadomości przez osobę atakującą, gdy nie została ona wybrana jednolicie na N (w rzeczywistości, gdy m jest wybrane z małego zakresu, łatwo jest podstawić Z przekonać się, że osoba atakująca może obliczyć m z klucza publicznego i szyfrogram). Ponadto nie wyklucza, że atakujący może poznać częściowe informacje o wiadomości, nawet jeśli jest ona jednolita (w rzeczywistości wiadomo, że jest to możliwe). Co więcej, zwykłe szyfrowanie RSA jest deterministyczne i dlatego musi być niepewne, jak już omówiliśmy w sekcji 11.2.1.

Więcej ataków na zwykły RSA

Zauważaliśmy już, że zwykłe szyfrowanie RSA nie jest zabezpieczone CPA. Niemniej jednak może pojawić się pokusa użycia zwykłego RSA do szyfrowania „losowych wiadomości” i/lub w sytuacjach, w których akceptowalny jest wyciek kilku fragmentów informacji o wiadomości. Ogólnie ostrzegamy przed tym i przedstawiamy tutaj tylko kilka przykładów tego, co może pójść nie tak.

(Niektóre z poniższych ataków zakładają e = 3. W niektórych przypadkach ataki można rozszerzyć, przyjmniej częściowo, na większe e; w każdym razie, jak zauważono w sekcji 8.2.4, często stosuje się ustalenie e = 3 w praktyce. Ataki należy traktować jako wykazanie, że Konstrukcja 11.26 jest nieodpowiednia, a nie jako wskazanie, że ustalenie e = 3 jest koniecznie złym wyborem.)

Kwadratowa poprawa odzyskiwania m.in. Ponieważ zwykłe szyfrowanie RSA jest deterministyczne, wiemy, że jeśli m < B, to osoba atakująca może określić m na podstawie zaszyfrowanego tekstu $c = [me \bmod N]$ w czasie $O(B)$ za pomocą ataku brute-force opisanego w sekcji 11.2.1. Można jednak mieć nadzieję, że można zastosować zwykłe szyfrowanie RSA, jeśli B jest duże, tj. jeśli wiadomość jest wybrana z dość dużego zestawu wartości. Jednym z możliwych scenariuszy, w których może to nastąpić, jest kontekst szyfrowania hybrydowego (patrz sekcja 11.3), gdzie „wiadomość” jest losowym n-bitowym kluczem, a więc $B = 2^n$. Niestety istnieje sprytny atak, który z dużym prawdopodobieństwem przywraca m w czasie mniej więcej $O(\sqrt{B})$. Może to mieć znaczącą różnicę w praktyce: atak trwający 280 razy (powiedzmy) jest niewykonalny, ale atak trwający 240 razy jest stosunkowo łatwy do przeprowadzenia.

Opis ataku podano w Algorytmie 11.28. W naszym opisie 2, 1) oznaczają pewną stałą stałą zakładamy $B = 2^n$ i niech $a = \frac{1}{r}$ (patrz poniżej).

W złożoności czasowej algorytmu dominuje czas potrzebny na posortowanie par $2an$ (r, xr); można to zrobić w czasie $O(n \cdot 2an)$. Wyszukiwanie binarne jest używane w przedostatnim wierszu, aby sprawdzić, czy istnieje $r \leq xr = [s \bmod N]$.

Teraz naszkicujemy, dlaczego atak przywraca m z dużym prawdopodobieństwem. Niech $c =$

ALGORYTM 11.28 Atak na zwykłe szyfrowanie RSA

Wejście: Klucz publiczny N, e ; szyfrogram c
 Wynik: $m < 2$ takie, $\exists m \equiv c \pmod{N}$

```

set T := 2an
dla r = 1 do T :
    xr := [c/re mod N]
    sortuj pary {(r, xr)} dla s = 1 do T przez ich drugi składnik
    jeśli xr == [s e mod N] dla pewnego
        r powrotu [r · s mod N]
    
```

me mod N. Przy odpowiednim wyborze $a >$ można wykazać, że jeśli m jest jednolitą n-bitową liczbą całkowitą, to z dużym prawdopodobieństwem istnieje r, s z an, dla którego $m = r \cdot s$. bitowym, to z (Na przykład, jeśli $n = 64$, a więc m jest $1 < r \cdot s \leq 2$ losowym ciągiem 64-prawdopodobieństwem 0,35 istnieje r, s o długości co najwyżej 34 bitów, takie że $m = r \cdot s$. Zobacz szczegółowe informacje można znaleźć w odnośnikach na końcu rozdziału.) Zakładając, że tak jest, powyższy algorytm znajduje m, ponieważ

$$\text{do } ja = (r \cdot s) \text{ mi} \quad e = r \cdot e \cdot s \pmod{N},$$

i tak $xr = c/re = s e \pmod{N}$ z $r, s < T$.

Szyfrowanie krótkich wiadomości za pomocą małych liter e. Poprzedni atak pokazuje, jak odzyskać wiadomość m, o której wiadomo, że jest mniejsza niż pewna związana B w czasie mniej więcej $O(\sqrt{B})$. Tutaj pokazujemy, jak zrobić to samo w czasie $\text{pol}(N)$, jeśli $B = N^{1/e}$ (gdzie oznacza to pierwiastek eth z N jako liczby rzeczywistej).

Atak opiera się na obserwacji, że gdy $m < N^{1/e}$, podniesienie m do potęgi eth modulo N nie powoduje redukcji modułowej; tj. $[m \pmod{N}]$ jest równe liczbie całkowitej $m^e \pmod{N}$. Oznacza to, że biorąc pod uwagę szyfrogram $c = [m \pmod{N}]$, $1/e$ w stosunku do liczb całkowitych (tzn. poprzez obliczenie $m := c \pmod{N}$); można to łatwo atakujący nie może określić m zrobić w czasie $\text{pol}(c) = \text{pol}(N)$, ponieważ znalezienie pierwiastków eth jest łatwe na liczbach całkowitych i trudne tylko podczas pracy z mod N.

W przypadku małego e oznacza to poważną słabość zwykłego szyfrowania RSA. Na przykład, jeśli przyjmiemy $e = 3$ i założymy, że $N = 1024$ bity, wówczas atak zadziała nawet wtedy, gdy m jest jednolitą 300-bitową liczbą całkowitą; to po raz kolejny wyklucza bezpieczeństwo zwykłego RSA, nawet jeśli jest używany jako część hybrydowego schematu szyfrowania.

Szyfrowanie częściowo znanej wiadomości. Atak ten można postrzegać jako uogólnienie poprzedniego. Zakłada, że nadawca szyfruje wiadomość, której część jest znana (coś, co nie powinno skutkować atakiem przy zastosowaniu bezpiecznego schematu szyfrowania). Tutaj opieramy się na potężnym wyniku Coppersmitha, który stwierdzamy bez dowodu:

TWIERDZENIE 11.29 Niech $p(x)$ będzie wielomianem stopnia e. Wtedy w czasie $\text{pol}(N, e)$ można znaleźć wszystkie m takie, że $p(m) = 0 \pmod{N}$ i $|m| \leq N^{1/e}$.

Ze względu na zależność czasu działania od e, atak jest praktyczny tylko dla małych e. W dalszej części założymy, że e = 3 dla konkretności.

Założymy, że nadawca szyfruje wiadomość $m = m_1 m_2$ do odbiorcy za pomocą klucza publicznego N, 3, gdzie znana jest pierwsza część m_1 wiadomości, ale druga część m_2 nie. Dla konkretności, powiedzmy, że m_2 ma długość k bitów, więc $m = B \cdot m_1 + m_2$ gdzie pozwalamy $B = 2^k$. Biorąc pod uwagę wynikowy tekst zaszyfrowany $c = [(m_1 m_2)^3 \bmod N]$, podsłuchujący może zdefiniować $\hat{c} = (2k \cdot m_1 + x)^3 \bmod c$, wielomian sześcienny. Ten wielomian $p(x)$, mający m_2 jako pierwiastek (modulo N) i $|m_2| < B$. Twierdzenie 11.29 implikuje zatem, że atakujący może efektywnie obliczyć m_2 , jeśli $B = N^{1/3}$. Podobny atak działa, gdy znane jest m_2 , ale m_1 nie.

Szyfrowanie powiązanych wiadomości.⁵ W tym ataku zakłada się, że nadawca szyfruje dwie powiązane wiadomości do tego samego odbiorcy (co nie powinno skutkować atakiem w przypadku stosowania bezpiecznego schematu szyfrowania). Założymy, że nadawca szyfruje zarówno m , jak i $m + \delta$ do odbiorcy za pomocą klucza publicznego N, e, gdzie przesunięcie δ jest znane, ale m nie. Mając dwa szyfrogramy $c_1 = [m \bmod N]$ i $c_2 = [(m + \delta) \bmod N]$, podsłuchujący może zdefiniować dwa wielomiany $f_1(x) = c_1$ i $f_2(x) = (x + \delta) \bmod N$, c_2 (módulo N), każdy stopnia e.

Zauważ, że $x = m$ jest pierwiastkiem (modulo N) obu wielomianów, a zatem składnik liniowy ($x - m$) jest czynnikiem obu. Zatem, jeśli największy wspólny dzielnik $f_1(x)$ i $f_2(x)$ (jako wielomiany nad wspólnym dzielnikiem N) jest liniowy, ujawni m. Najwspanialszy Z można obliczyć w czasie $\text{pol}(N, e)$ stosując algorytm podobny do tego z Załącznika B.1.2; zatem, ten atak jest wykonalny dla małych e.

Wysyłanie tej samej wiadomości do wielu odbiorców.⁶ Nasz ostateczny atak zakłada, że nadawca szyfruje tę samą wiadomość do wielu odbiorców (coś, co po raz kolejny nie powinno skutkować atakiem przy użyciu bezpiecznego schematu szyfrowania). Niech e = 3 i powiedzmy, że ta sama wiadomość m jest szyfrowana do trzech różnych stron posiadających klucze publiczne, odpowiednio, $pk_1 = N_1, 3$, $pk_2 = N_2, 3$ i $pk_3 = N_3, 3$. Założymy, że $\gcd(N_i, N_j) = 1$ dla różnych i, j; jeśli nie, wówczas co najmniej jeden z modułów można natychmiast rozłożyć na czynniki i komunikat m można łatwo odzyskać.

Podsłuchujący widzi

$$c_1 = [m \bmod N_1], c_2 = [m \bmod N_2] \text{ i } c_3 = [m \bmod N_3].$$

Niech $N = N_1 N_2 N_3$. Rozszerzona wersja chińskiego twierdzenia o resztach mówi, że istnieje unikalna nieujemna liczba całkowita $\hat{c} < N$ taka, że

$$\begin{aligned}\hat{c} &\equiv c_1 \pmod{N_1} \\ \hat{c} &\equiv c_2 \pmod{N_2} \\ \hat{c} &\equiv c_3 \pmod{N_3}.\end{aligned}$$

⁵Atak ten opiera się na algebrze nieco wykraczającej poza to, co omówiliśmy w tej książce.

⁶Atak ten opiera się na chińskim twierdzeniu o resztach przedstawionym w podrozdziale 8.1.5.

Co więcej, stosując techniki podobne do tych pokazanych w sekcji 8.1.5, możliwe jest efektywne obliczenie \hat{c} , biorąc pod uwagę klucze publiczne i powyższe teksty zaszyfrowane. Na koniec zauważmy, że m^3 spełnia powyższe równania, a $m^3 < N$ ponieważ $m < \min\{N_1, N_2, N_3\}$. Podobnie jak w poprzednim ataku, oznacza to, że $\hat{c} = m^3$ przez liczby całkowite (tzn. bez redukcji modułowej), zatem komunikat m można odzyskać, obliczając całkowity pierwiastek sześcienny z \hat{c} .

11.5.2 Wyściełane RSA i PKCS #1 v1.5

Chociaż zwykły RSA nie jest bezpieczny, sugeruje jedno ogólne podejście do szyfrowania klucza publicznego w oparciu o problem RSA: aby zaszyfrować wiadomość m przy użyciu klucza publicznego N, e , najpierw mapuj m na element $\hat{m} \in Z_N$; następnie oblicz szyfrogram $c = [\hat{m}^e \bmod N]$. Aby odszyfrować tekst zaszyfrowany c , odbiorca oblicza $\hat{m}^{\frac{1}{e}} = [c^{\frac{1}{e}} \bmod N]$, a następnie odzyskuje oryginalną wiadomość m . Aby odbiorca mógł odzyskać wiadomość, mapowanie wiadomości na elementy musi być (efektywnie) odwracalne. Aby

schemat zgodny z podejściem Z_N miał nadzieję być bezpieczny pod względem CPA, mapowanie musi być losowe, aby szyfrowanie nie było deterministyczne. Jest to oczywiście warunek konieczny, ale niewystarczający, a bezpieczeństwo schematu szyfrowania zależy w decydującym stopniu od zastosowanego konkretnego mapowania.

Jedną z prostych implementacji powyższego pomysłu jest losowe uzupełnienie wiadomości przed zaszyfrowaniem. Oznacza to, że aby odwzorować wiadomość m (postrzeganą jako ciąg bitów) na element Z_N , nadawca wybiera jednolity ciąg bitów $r \in \{0, 1\}$ (dla niektórych odpowiednich r) i ustawia $\hat{m} := rm$; wynikową wartość można oczywiście zinterpretować jako liczbę całkowitą $m \in Z_N$ i to odwzorowanie jest wyraźnie odwracalne. Zobacz Budowa 11.30. (Granice (n) i długość m zapewniają, że liczba całkowita \hat{m} jest mniejsza niż N .)

Konstrukcja jest parametryzowana wartością, która określa długość zastosowanego losowego wypełnienia. Bezpieczeństwo programu zależy od r . Istnieje oczywisty atak brute-force na schemat, który działa w czasie 2^r , więc jeśli jest za krótki (w szczególności jeśli $(n) = O(\log n)$), schemat jest niepewny. Z drugiej strony w następnej sekcji pokażemy (zasadniczo), że jeśli dopełnienie jest tak duże, jak to możliwe, a m jest tylko pojedynczym bitem, wówczas możliwe jest udowodnienie bezpieczeństwa w oparciu o założenie RSA. W przypadkach pośrednich sytuacja jest mniej jasna: dla pewnych zakresów nie możemy udowodnić bezpieczeństwa w oparciu o założenie RSA, ale nie są też znane żadne ataki z czasem wielomianowym. Odkładamy dalszą dyskusję do czasu, aż zajmiemy się PKCS #1 v1.5 w następnej kolejności.

RSA PKCS #1 wersja 1.5. Standard RSA Laboratories Public-Key Cryptography Standard (PKCS) nr 1, wersja 1.5, wydany w 1993 r., wykorzystuje wariant rozszerzonego szyfrowania RSA. Dla klucza publicznego $pk = N, e$ w zwykłej postaci, niech k oznacza długość N w bajtach; tj. k jest liczbą całkowitą spełniającą $28(k-1) \leq N < 2^k$. Zakłada się, że wiadomości m do zaszyfrowania⁸⁵ są wielokrotnością 8 bitów i mogą mieć długość od jednego do $k - 11$ bajtów. Szyfrowanie wiadomości

BUDOWA 11.30

Niech GenRSA będzie jak poprzednio i niech będzie funkcją $z(n) = 2n - 4$ dla wszystkich n . Zdefiniuj schemat szyfrowania klucza publicznego w następujący

- sposób:
- Gen: na wejściu n oznaczonym GenRSA(1^n), aby otrzymać (N, e, d) . Wyprowadź klucz publiczny $pk = N$, e i klucz prywatny $sk = N, d$.
 - Enc: na wejściu klucz publiczny $pk = N, e$ i wiadomość wybierającą jednolity串 $r \in \{0, 1\}^{z(n)}$ i N .
 $m \in \{0, 1\}^{N(n)-2}$, Wyprowadź szyfrrogram
zinterpretuj $\hat{m} := rm$ jako element Z

$$\text{do: } [\hat{m}^m \bmod N].$$

- Dec: na wejściu klucz prywatny $sk = N, d$ i szyfrrogram $c \in Z$
obliczać
 $m^{\hat{m}} := [c^d \bmod N],$
wypisz $N - (n) - 2$ najmniej znaczące bity \hat{m} .

Wyświetlany schemat szyfrowania RSA.

m , czyli długość D-bajtów, jest obliczana jako

$$[(0x000x02r0x00m) \bmod N],$$

gdzie r jest losowo wygenerowanym ciągiem znaków ($k-D-3$ -bajtów, w którym żaden z bajtów nie jest równy 0x00). (Ten ostatni warunek umożliwia jednoznaczne odtworzenie wiadomości po odszyfrowaniu.) Należy zauważyć, że maksymalna dozwolona długość m zapewnia, że długość r wynosi co najmniej 8 bajtów.

Niestety, PKCS #1 v1.5, jak określono, nie jest bezpieczny pod względem CPA, ponieważ pozwala na użycie zbyt krótkiego losowego dopełnienia. Najlepiej ilustruje to pokazanie, że osoba atakująca może określić początkową część wiadomości, o której wiadomo, że zawiera wiele końcowych zer. Dla uproszczenia powiedzmy $m = b_0 \cdots 0$ gdzie $b_i \in \{0, 1\}$ wynosi

nieznane, a m jest tak długie, jak to możliwe (więc $L = 8 \cdot (k - 11) + 1$). Szyfrowanie m daje zaszyfrowany tekst c

$$c = (0x000x02r0x00b0 \cdots 0)e \bmod N.$$

Osoba atakująca może obliczyć $c = c/(2L) e \bmod N$; zauważ to

$$c = \frac{0x000x02r0x00b0 \cdots 0}{10 \cdots 0} = (0x000x02r0x00b) \bmod N.$$

Liczba całkowita $0x02r0x00b$ ma długość 75 bitów (należy pamiętać, że $0x02 = 0000 0010$ i wszystkie bity 0 wyższego rzędu nie są liczone), dlatego osoba atakująca może teraz zastosować „atak za pomocą krótkiej wiadomości” lub atak oparty na szyfrowaniu częściowo znanej wiadomości, z poprzedniej sekcji. Aby uniknąć tych ataków, musimy przyjąć r o długości co najmniej N/e . Jednak nawet jeśli e jest duże, „atak polegający na ulepszeniu kwadratowym” z poprzedniej sekcji pokazuje, że r można odzyskać z dużym prawdopodobieństwem w czasie w przybliżeniu $2r/2$.

Jeśli wymusimy, aby r było mniej więcej o połowę krótsze od N i odpowiednio zmniejszymy maksymalną długość wiadomości, wówczas rozsądne jest przypuszczenie, że schemat szyfrowania w PKCS #1 v1.5 jest zabezpieczony CPA. (Podkreślamy jednak, że nie jest znany żaden dowód bezpieczeństwa oparty na założeniu RSA.) Niemniej jednak, z powodu poważnego ataku wybranego tekstu zaszyfrowanego na schemat, opisanego pokrótko w sekcji 11.5.5, nowsze wersje PKCS # Wprowadzono 1 normę, która powinna być stosowana zamiast niej.

11.5.3 * Szyfrowanie zabezpieczone CPA bez losowych Oracle

W tej sekcji pokazujemy schemat szyfrowania, który można udowodnić, że jest bezpieczny pod względem CPA w oparciu o założenie RSA. Zaczynamy od opisu konkretnego predykatu twardego (patrz sekcja 7.1.3) dla problemu RSA, a następnie pokazujemy, jak używać tego predykatu twardego do szyfrowania pojedynczego bitu. Następnie rozszerzamy ten schemat, aby uzyskać KEM.

Schematy opisane w tej sekcji mają głównie znaczenie teoretyczne i nie są stosowane w praktyce. Dzieje się tak dlatego, że są one mniej wydajne niż alternatywne konstrukcje oparte na RSA, których bezpieczeństwo można udowodnić w modelu losowej wyroczni (por. sekcja 5.5). Przykłady takich schematów szyfrowania zobaczymy w kolejnych sekcjach.

Twardy predykat problemu RSA. Mówiąc luźno, założenie RSA mówi, że biorąc pod uwagę N , e i $[x \text{ e mod } N]$ (dla x wybranego jednolicie z Z_N), odzyskanie x jest niemożliwe. Samo to nie mówi nic o trudnościach obliczeniowych związanych z obliczeniem określonej informacji o x . Czy możemy wyizolować jakiś konkretny fragment informacji o x , który jest trudny do obliczenia z N , e i $[x \text{ e mod } N]$? Pojęcie twardego predykatu dokładnie oddaje ten wymóg. (Predykaty twarde zostały wprowadzone w sekcji 7.1.3.

Fakt, że założenie RSA daje rodzinę jednokierunkowych permutacji, omówiono w podrozdziale 8.4.1. Jednakże nasze leczenie jest tutaj niezależne.)

Okazuje się, że najmniej znaczący bit x , oznaczony $\text{lsb}(x)$, jest twardym predykatem problemu RSA.

Zdefiniuj następujący eksperyment dla danego algorytmu GenRSA (z rozszerzeniem zwykłego zachowania) i algorytmem A:

Twardy eksperyment predykatowy RSA RSA-lsbA,GenRSA(1n):

1. Uruchom GenRSA(1n), aby otrzymać (N, e) ,
2. wybierz uniform $x \in Z_N$.
3. A. oblicz $y := [x \text{ e mod } N]$.
ma dane N, e, y i wprowadza bit b .
4. Wynik eksperymentu wynosi 1 wtedy i tylko wtedy, gdy $\text{lsb}(x) = b$.

Zauważ, że $\text{lsb}(x)$ jest bitem jednolitym, gdy $x \in Z_N$ jest jednolity. A może odgadnąć $\text{lsb}(x)$ z prawdopodobieństwem $1/2$, po prostu wysyłając jednolity bit b .

Poniższe twierdzenie stwierdza, że jeśli problem RSA jest trudny, to nie jest efektywny

algorytm A może zrobić znacznie lepiej; tj. najmniej znaczący bit jest twardym predykatem permutacji RSA.

TWIERDZENIE 11.31 Jeśli problem RSA jest trudny w porównaniu z GenRSA , to dla wszystkich probabilistycznych algorytmów A w czasie wielomianowym istnieje funkcja zaniedbywalna , taka że $\Pr[\text{RSA-lsbA,GenRSA}(n) = 1] \geq \negl(n)$. 12

Pełny dowód tego twierdzenia wykracza poza zakres tej książki. Jednakże zapewniamy pewną intuicję dla twierdzenia, szkicując dowód słabszego wyniku: że założenie RSA implikuje $\Pr[\text{RSA-lsbA,GenRSA}(n) = 1] < 1$ dla wszystkich probabilistycznych wielomianów-czasów A. Aby to udowodnić, pokazują, że wydajny algorytm, który zawsze poprawnie oblicza lsb(r) z N, e i [re mod N], może zostać użyty do wydajnego odzyskania x (w całości) z N, e i [x e mod N].

Ustal N i e i niech A będzie takim algorytmem, że $A([re \bmod N]) = \text{lsb}(r)$. Biorąc pod uwagę N, e i y = [x e mod N], odzyskamy bity x jeden po drugim, od najmniejszego do najbardziej znaczącego. Aby określić lsb(x), po prostu uruchamiamy A(y). Tam są teraz dwa przypadki:

Przypadek 1: $\text{lsb}(x) = 0$. Zauważ, że $y/2 \equiv (x/2)e \bmod N$, a ponieważ x jest parzyste (tzn. $\text{lsb}(x) = 0$), 2 dzieli liczbę całkowitą x. Zatem $x/2$ jest po prostu przesunięciem bitowym x w prawo, a $\text{lsb}(x/2)$ jest równe $2\text{sb}(x)$, czyli drugim najmniej znaczącym bitem x. Możemy więc otrzymać $2\text{sb}(x)$ obliczając $y := [y/2 e \bmod N]$, a następnie uruchamiając A(y).

Przypadek 2: $\text{lsb}(x) = 1$. Tutaj $[x/2 \bmod N] = (x + N)/2$. Zatem $\text{lsb}([x/2 \bmod N])$ równa się $2\text{sb}(x + N)$; ta ostatnia jest równa $1 - 2\text{sb}(N) - 2\text{sb}(x)$ (na drugiej pozycji mamy bit przeniesienia, ponieważ zarówno x, jak i N są nieparzyste). Zatem jeśli obliczymy $y := [y/2 e \bmod N]$, to $2\text{sb}(x) = A(y) - 1 - 2\text{sb}(N)$.

Kontynuując w ten sposób, możemy odzyskać wszystkie bity x.

Szyfrowanie jednego bitu. Możemy użyć twardego predykatu wskazanego powyżej, aby zaszyfrować pojedynczy bit. Pomyśl jest prosty: aby zaszyfrować wiadomość $m \in \{0, 1\}$, nadawca wybiera uniform r $\in \mathbb{Z}$ pod warunkiem, że $\text{lsb}(r) = m$; szyfrogram to $c := [re \bmod N]$. Zobacz Konstrukcja 11.32.

TWIERDZENIE 11.33 Jeśli problem RSA jest trudny w porównaniu z GenRSA, wówczas Konstrukcja 11.32 jest zabezpieczona CPA.

DOWÓD Niech Π oznacza konstrukcję 11.32. Udowodnimy, że Π ma nieroróżnicjalne szyfrowanie w obecności podsłuchującego; według Propozycji 11.3 oznacza to, że jest on bezpieczny dla CPA.

Niech A będzie probabilistycznym przeciwnikiem wielomianowym w czasie. Bez utraty mocy

BUDOWA 11.32

Niech GenRSA będzie jak zwykle i zdefiniuje schemat szyfrowania klucza publicznego w następujący sposób:

- Gen: na wejściu $1n$, uruchom GenRSA($1n$), aby otrzymać (N, e, d) . Wyprowadź klucz publiczny $pk = N$, e i klucz prywatny $sk = N, d$.
- Enc: na wejściu klucz publiczny $pk = N, e$ i komunikat $m \in \{0, 1\}$, N z zastrzeżeniem ograniczenia $lsb(r) = m$. wybierz mundur $r \in Z^D$
- Wypisz zaszyfrowany tekst $c := [re \bmod N]$.
- Dec: na wejściu klucz prywatny $sk = N, d$ i tekst zaszyfrowany c , com-mod N i na wyjściu $lsb(r)$.
pute $r := [ok]$

Szyfrowanie jednobitowe przy użyciu twardego predykatu dla RSA.

W eksperymencie PubKeav $A, \Pi(n)$ możemy przyjąć $m_0 = 0$ i $m_1 = 1$. Więc

$$\Pr[\text{PubKeav } A, \Pi(n) = 1] = \frac{1}{2} \cdot \Pr[A(N, e, c) = 0 \mid c \text{ jest szyfrowaniem } 0] + \frac{1}{2} \cdot \Pr[A(N, e, c) = 1 \mid c \text{ jest szyfrowaniem } 1].$$

Rozważ uruchomienie A w eksperymencie RSA-lsb. Zgodnie z definicją,

$$\Pr[\text{RSA-lsb } A, \text{GenRSA}(n) = 1] = \Pr[AN, e, [r \bmod N] = lsb(r)],$$

gdzie r jest jednolite w Z^N . Ponieważ $\Pr[lsb(r) = 1] = 1/2$, mamy

$$\Pr[\text{RSA-lsb } A, \text{GenRSA}(n) = 1] = \frac{1}{2} \cdot \Pr[AN, e, [re \bmod N] = 0 \mid lsb(r) = 0] + \frac{1}{2} \cdot \Pr[AN, e, [re \bmod N] = 1 \mid lsb(r) = 1].$$

Zauważając, że szyfrowanie $m \in \{0, 1\}$ odpowiada dokładnie wyborowi uniform r z zastrzeżeniem ograniczenia $lsb(r) = m$, widzimy, że

$$\Pr[\text{PubKeav } A, \Pi(n) = 1] = \Pr[\text{RSA-lsb } A, \text{GenRSA}(n) = 1].$$

Twierdzenie 11.31 implikuje zatem, że istnieje pomijalna funkcja negl taka, że

$$\Pr[\text{PubKeav } A, \Pi(n) = 1] = \frac{1}{2} \cdot \text{zaniebywanie}(n).$$

zgodnie z życzeniem. ■

Budowa KEM-u. Pokażemy teraz, jak rozszerzyć Konstrukcję 11.32, aby uzyskać KEM o długości klucza n . Naiwnym sposobem byłoby po prostu wybranie jednolitego, n -bitowego klucza k , a następnie zaszyfrowanie bitów k jeden po drugim

używając n wywołań Construction 11.32. W rezultacie powstanie dość długi tekst zaszyfrowany składający się z n elementów ZN .

Lepszym podejściem jest wielokrotne zastosowanie przez nadawcę permutacji RSA (mianowicie podniesienia do potęgi eth modulo N), zaczynając od początkowej, jednolitej wartości c1. Oznacza to, że nadawca będzie sukcesywnie obliczał c (ce 1) e $\stackrel{\text{mod}}{\equiv}$, następnie mod $= \text{do } t^{m^2}$, tak dalej, aż do c $\stackrel{\text{mod}}{\equiv}$ (wszystkie modulo N). Wartość końcowa $[c \text{ e } 1 \stackrel{\text{mod }}{\equiv} N]$ będzie szyfrogramem i sekwencją bitów $\text{lsb}(c1), \text{lsb}(c \text{ e } 1), \dots, \text{lsb}(c \text{ e } n)$ jest kluczem. Aby odszyfrować zaszyfrowany tekst c, odbiorca po prostu odwraca ten proces, sukcesywnie obliczając c $\stackrel{\text{mod }}{\equiv} (C^D)^{-1} \text{ (po dwukrotnie wszystkie modulo N)}$ aż do c używanego aby odzyskać wartość początkową $c1 = c, \stackrel{\text{mod }}{\equiv} N$ nadawcę. Po odzyskaniu c1, odbiornik może następnie ponownie obliczyć $c_1, \dots, c_n \stackrel{\text{mod }}{\equiv} N$ i zdobądź klucz.

Możliwe jest skuteczniejsze wdrożenie deszyfrowania, wykorzystując fakt, że N . W momencie odbiorca zna kolejność grupy Z, odbiorca może wstępnie generowania klucza plik obliczyć $d := [d \text{ n mod } \varphi(N)]$ i zapisać d jako część swojego klucza prywatnego. Zamiast obliczać n kolejnych wykładni modułowych, c $c1 := [c \text{ mod } N]$. To oczywiście działa, ponieważ

$$d^2 \stackrel{\text{mod }}{\equiv} \dots \text{ aby uzyskać } c1, \text{ odbiorca może zamiast tego bezpośrednio obliczyć } d$$

$$- \stackrel{\text{mod }}{\equiv} N \quad [d \text{ n mod } \varphi(N)] \quad d = \text{do } \mod N.$$

Powyższe jest formalnie opisane jako Konstrukcja 11.34.

BUDOWA 11.34

Niech GenRSA będzie jak zwykle i zdefiniuje KEM w następujący sposób:

- Gen: na wejściu $1n$, uruchom GenRSA($1n$), aby otrzymać (N, e, d) . Następnie puste $d := [d \text{ n com-mod } \varphi(N)]$ (zauważ, że $\varphi(N)$ można obliczyć z (N, e, d) lub uzyskane w trakcie działania GenRSA). Wyjście $pk = N, e$ i $sk = N, re$.
- Encaps: na wejściu $pk = N, e$ i $1n$, wybierz mundur $c1 \in Z_N$. Wtedy dla $i = 1, \dots, n$, zrób:
 - Oblicz $ki := \text{lsb}(ci)$.
 - Oblicz $ci+1 := [c \text{ e mod } N]$.

Wyprowadź szyfrogram $cn+1$ i klucz $k = k1 \dots kn$.

- Decaps: na wejściu $sk = N, d$ i zaszyfrowanym tekście c, oblicz $c1 := [c^D \text{ mod } N]$. Wtedy dla $i = 1, \dots, n$, zrób:
 - Oblicz $ki := \text{lsb}(ci)$.
 - Oblicz $ci+1 := [c \text{ e mod } N]$.

Wyprowadź klucz $k = k1 \dots kn$.

KEM wykorzystujący twardy predykat dla RSA.

Konstrukcja przypomina podejście zastosowane do skonstruowania generatora pseudolosowego z permutacji jednokierunkowej pod koniec rozdziału 7.4.2. Jeśli pozwolimy f oznaczać permutację RSA względem jakiegoś klucza publicznego

$f(x) = [x \text{ mod } N]$, wówczas CPA-bezpieczeństwo konstrukcji 11.34 wynosi równoważne pseudolosowości $\text{lsb}(f)$ na wartości $c^{n-1} (c1), \dots, \text{lsb}(c1)$ nawet uwarunkowane $c = f(n(c1))$. To z kolei można udowodnić za pomocą Twierdzenia 11.31 i technik z rozdziału 7.4.2. (Jedyna różnica polega na tym, że w rozdziale 7.4.2 wartość $f(n(c1))$ sama była jednolitym n-bitowym ciągiem znaków, podczas gdy tutaj jest jednolitym elementem Z N. Pseudolosowość kolejnych twardych predykatów jest niezależna od dziedziny f.)

Podsumowując:

TWIERDZENIE 11.35 Jeśli problem RSA jest trudny w porównaniu z GenRSA, wówczas Konstrukcja 11.34 jest KEM bezpiecznym dla CPA.

Efektywność. Konstrukcja 11.34 jest w miarę wydajna. Konkretnie założymy, że $n = 128$, moduł RSA N ma długość 2048 bitów, a wykładek publiczny e wynosi 3, więc potęgowanie do potęgi e modulo N można obliczyć za pomocą dwóch mnożeń modułowych. (Patrz dodatek B.2.3.) Szyfrowanie wymaga zatem $2n = 256$ mnożeń modułowych. Odszyfrowanie można przeprowadzić za pomocą jednego pełnego potęgowania modułowego (kosztem około $1.5 \cdot 2048 = 3072$ mnożeń modułowych) plus dodatkowe 256 mnożeń modułowych. Koszt deszyfrowania jest zatem tylko o około 8% mniej efektywny niż w przypadku zwykłego schematu szyfrowania RSA. Natomiast szyfrowanie jest znacznie droższe niż zwykły RSA, ale w wielu aplikacjach czas deszyfrowania jest znacznie ważniejszy (ponieważ może być zaimplementowany przez serwer, który wykonuje tysiące deszyfracji jednocześnie).

11.5.4 OAEP i RSA PKCS #1 v2.0

Jak dotąd nie rozważaliśmy schematów szyfrowania opartych na RSA, które są zabezpieczone CCA. Najpierw pokazujemy, że wszystkie schematy szyfrowania oparte na RSA, które widzieliśmy do tej pory, są podatne na ataki z użyciem wybranego tekstu zaszyfrowanego.

Zwykłe szyfrowanie RSA. Zwykły RSA nie jest nawet bezpieczny pod względem CPA. Zapewnia jednak, że jeśli m jest jednolite, wówczas atakujący, który podsłuchuje szyfrowanie $c = [me \text{ mod } N]$ m w odniesieniu do klucza publicznego N, e, nie może odzyskać m. Nawet ta słaba gwarancja nie obowiązuje już w środowisku, w którym możliwe są ataki wybranym tekstem zaszyfrowanym. Podobnie jak w przypadku szyfrowania El Gamala, jest to konsekwencja faktu, że zwykły RSA jest plastyczny: mając szyfrowanie $c = [me \text{ mod } N]$ nieznanej wiadomości m, łatwo jest wygenerować zaszyfrowany tekst c będący szyfrowaniem z $[2m \text{ mod } N]$ poprzez ustalenie

$$\begin{aligned} c &:= [2e \cdot c \text{ mod } N] \\ &= 2e \cdot me = (2m) e \text{ mod } N. \end{aligned}$$

RSA PKCS #1 wersja 1.5. Wyścianie szyfrowanie RSA, które przypuszczalnie jest bezpieczne według CPA w celu prawidłowego ustawnienia parametrów, jest podatne na zasadniczo ten sam atak, co zwykłe szyfrowanie RSA. Istnieje jednak również bardziej interesujący atak z użyciem wybranego tekstu zaszyfrowanego na szyfrowanie PKCS #1 v1.5, który w przeciwnieństwie do ataku przedstawionego powyżej nie wymaga pełnego dostępu do wyroczni deszyfrującej; wymaga jedynie dostępu do „częściowej” wyroczni deszyfrującej, która wskazuje, czy odszyfrowanie jakiegoś tekstu zaszyfrowanego zwraca błąd. To sprawia, że atak jest o wiele bardziej praktyczny, ponieważ można go przeprowadzić zawsze, gdy atakujący jest w stanie odróżnić zachowanie odbiorcy po pomylnym odszyfrowaniu od jego zachowania po niepowodzeniu odszyfrowania, jak w przypadku ataku padding-oracle pokazanego w sekcji 3.7.2.

Przypomnijmy, że schemat szyfrowania kluczem publicznym zdefiniowany w standardzie PKCS #1 v1.5 wykorzystuje wariant dopełnianego szyfrowania RSA, w którym dopełnienie odbywa się w określony sposób. W szczególności dwa najważniejsze bajty dopełnianej wiadomości mają zawsze wartość 0x000x02. Podczas deszyfrowania odbiorca powinien sprawdzić, czy dwa najważniejsze bajty odpowiadają tym wartościom, i zwrócić błąd, jeśli tak nie jest. W 1998 roku Bleichenbacher opracował atak z użyciem wybranego tekstu zaszyfrowanego, który wykorzystuje fakt, że to sprawdzenie jest przeprowadzane. Z grubsza, biorąc pod uwagę zaszyfrowany tekst c , który odpowiada uczciwemu szyfrowaniu jakiejś nieznanej wiadomości m w odniesieniu do klucza publicznego N , e , atak wielokrotnie wybiera jednolite $s \in [s \in [^m \text{ mod } N]]$ do odbiornika. Powiedz $c =$ tekst $c := [s \in [^m \text{ mod } N]]$ Gdzie

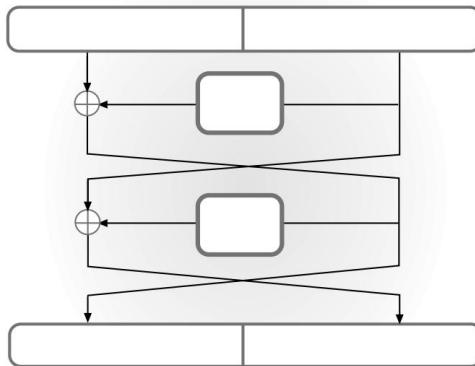
$$m^s = 0x000x02r0x00m,$$

jak określono w PKCS #1 v1.5. Następnie odszyfrowanie c da wynik pośredni $\hat{m} = [s \cdot m^s \text{ mod } N]$, a odbiorca zwróci błąd, chyba że dwa górne bajty \hat{m} mają dokładnie wartość 0x000x02. (Inne kontrole są również przeprowadzane, ale dla uproszczenia je ignorujemy.) Za każdym razem, gdy odszyfrowanie się powiedzie, osoba atakująca dowiaduje się, że dwa górne bajty $s \cdot m^s \text{ mod } N$ to 0x000x02, gdzie znane jest s . Szereg równań tego typu wystarczy, aby atakujący nauczył się m i odzyskał całą oryginalną wiadomość m .

KEM z zabezpieczeniem CPA. W sekcji 11.5.3 pokazaliśmy konstrukcję KEM, której bezpieczeństwo CPA można udowodnić w oparciu o założenie RSA. Konstrukcja ta jest również niezabezpieczona przed atakiem z wykorzystaniem wybranego tekstu zaszyfrowanego; szczegóły zostawiamy jako ćwiczenie.

RSA-OAEP

W tej sekcji badamy konstrukcję szyfrowania bezpiecznego CCA opartego na RSA przy użyciu optymalnego asymetrycznego dopełniania szyfrowania (OAEP). Powstały schemat RSA-OAEP jest zgodny z ideą (wykorzystaną także w podrozdziale 11.5.2) przyjęcia komunikatu m i przekształcenia go na element $\hat{m} \in [^m \text{ mod } N]$, a następnie niech $c = [\hat{m} \text{ mod } N]$ będzie tekstem zaszyfrowanym. Transformacja tutaj jednak jest



RYSUNEK 11.4: Mechanizm OAEP.

bardziej złożone niż wcześniej. Wersja RSA-OAEP została ujednoliciona jako część RSA PKCS #1 od wersji 2.0.

Niech $(n, k0(n), k1(n))$ będą funkcjami o wartościach całkowitych, gdzie $k0(n), k1(n) = \Theta(n)$ i takimi, że $(n + k0(n) + k1(n))$ jest mniejsza niż minimalna długość bitowa modułów wyjściowych przez GenRSA(1n). Ustal n i niech $= (n, k0 = k0(n),$ oraz $+k1 \in H : \{0, 1\}$) $k0$ będzie dwie funkcje mieszające, które będą modelowane jak $\Phi(n)$ i $\Lambda(n)$ dla losowej wyroczni $\{0, 1\}$ (Chociaż użycie więcej niż jednej losowej wyroczni nie zostało omówione w podrozdziale 5.5.1, jest to interpretowane w naturalny sposób.) Transformacja zdefiniowana przez OAEP to dwuokrągła sieć Feistela z G i H jako funkcjami okrągłymi; patrz rysunek 11.4. Szczegółowo dopełnianie wiadomości $m \in \{0, 1\}$ odbywa się w następujący sposób: najpierw ustawiamy $m := m0 k1$ i wybieramy uniform $r \in \{0, 1\}$

$k0$. Następnie oblicz

$$s := m \quad G(r) \quad \{0, 1\} \quad +k1, \quad t := r \quad H(s) \quad \{0, 1\} \quad k0,$$

i ustaw $\hat{m} := st$.

Aby zaszyfrować wiadomość m w odniesieniu do klucza publicznego N, e , nadawca generuje \hat{m} jak powyżej i wysyła zaszyfrowany tekst $c := \hat{m}e \bmod N$. (Zauważ, że \hat{m} , interpretowane jako liczba całkowita, jest mniejsze niż N ze względu na ograniczenia mod N NA, $k0, k1$.) Aby odszyfrować, odbiorca oblicza $\hat{m} := [c]^D$ i pozwala $st := \hat{m}m$ z s i t o odpowiednich długościach. Następnie odwraca sieć Feistela obliczając $r := H(s) - t$ i $m := G(r) - s$. Co ważne, odbiornik następnie sprawdza, czy wszystkie końcowe bity $k1$ mają wartość 0. Jeśli nie, szyfrrogram jest odrzucony i zwracany jest komunikat o błędzie. W przeciwnym razie najmniej znaczące zera $k1$ w m są odrzucone, a pozostałe bity m są wyprowadzane jako Ten . Proces jest opisany w Construction 11.36.

Dowód bezpieczeństwa CCA dla RSA-OAEP jest dość skomplikowany i nie podajemy go tutaj. Zamiast tego zapewniamy jedynie intuicję. Najpierw rozważ

BUDOWA 11.36

Niech GenRSA będzie jak w poprzednich sekcjach, a $+k1$ i $H: \{0, 1\}^k \times \{0, 1\}^k \rightarrow \{0, 1\}^k$ być zgodne z opisem. G : {0, 1} → {0, 1} → {0, 1} będą funkcjami. Skonstruuj schemat szyfrowania klucza publicznego w następujący sposób:

- Gen: na wejściu $1n$, uruchom GenRSA($1n$), aby otrzymać (N, e, d) . Klucz publiczny to N , e , a klucz prywatny to N, d .
- Enc: po wprowadzeniu klucza publicznego N , e i wiadomości $m \in \{0, 1\}^k$, $m := m0 \oplus k1$ i wybierz uniform $r \in \{0, 1\}^k$ ustawić. Następnie oblicz

$$s := m \oplus G(r), t := r \oplus H(s)$$

i ustaw $\hat{m} := st$. Wyprowadź szyfrogram $c := [\hat{m}e \bmod N]$.

- Dec: na wejściu klucz prywatny N, d i tekst zaszyfrowany $c \in \mathbb{Z}_N$, compute $\hat{m} := [c^d \bmod N]$. Jeśli $\hat{m} > +k0+k1$, wyjście \hat{m} . W przeciwnym razie $k0$. t i $m := G(r) \oplus k1$. Jeli $\hat{m} < 0$ wypisz $k1$ jako st z $s \in \{0, 1\}^k$, $r := H(s)$. Oblicz najmniej znaczące bity $k1$ m mają wartość 0, wyprowadź \hat{m} . W przeciwnym razie wypisz najbardziej znaczące bity \hat{m} .

Schemat szyfrowania RSA-OAEP.

Bezpieczeństwo CPA. Podczas szyfrowania nadawca dokonuje obliczeń

$$m := m0 \oplus k1, \quad s := m \oplus G(r), t := r \oplus H(s)$$

dla jednolitego r ; szyfrogram to $[(st) e \bmod N]$. Jeśli atakujący nigdy nie odpytuje r do G , to ponieważ modelujemy G jako funkcję losową, wartość $G(r)$ jest jednolita z punktu widzenia atakującego, a zatem m jest maskowane jednolitym ciągiem znaków, tak jak w przypadku jednorazowej podkładki schematu szyfrowania. Zatem jeśli atakujący nigdy nie zapyta r do G , wówczas żadne informacje o wiadomości nie wyciekną.

Czy atakujący może wysłać zapytanie r do G ? Należy zauważyć, że wartość r jest sama w sobie maskowana przez $H(s)$. Zatem atakujący nie ma żadnych informacji o r , chyba że najpierw zapyta s do H . Jeśli atakujący nie zapyta s do H , atakujący może mieć szczęście i tak zgadnąć r , ale jeśli odpowiednio ustawimy długość r (tj. $k0$) dugo, to prawdopodobieństwo tego jest znikome.

Zatem jedynym sposobem, aby atakujący mógł dowiedzieć się czegokolwiek o m , jest najpierw zapytanie s do H . Wymagałoby to od atakującego obliczenia s na podstawie (jednoformowego) tekstu zaszyfrowanego $[(st) e \bmod N]$. Zauważ, że obliczanie s z $[(st) e \bmod N]$ nie jest problemem RSA, który zamiast tego obejmuje obliczanie zarówno s , jak i t . Niemniej jednak dla prawidłowego ustawienia parametrów możemy użyć Twierdzenia 11.29, aby pokazać, że odzyskiwanie s umożliwia odzyskiwanie t w czasie wielomianowym, a więc odzyskiwanie s jest obliczeniowo niewykonalne, jeśli problem RSA jest trudny.

Argumentowanie, że bezpieczeństwo CCA wiąże się z dodatkowymi komplikacjami, ale podstawową ideą jest pokazanie, że każde zapytanie deszyfrujące-wyrocznia c wykonane przez atakującego należy do jednej z dwóch kategorii: albo atakujący uzyskał c poprzez legalne zaszyfrowanie jakiejś wiadomości m (w którym to przypadku atakujący nie uczy się niczego z zapytania deszyfrującego), w przeciwnym razie odszyfrowanie c zwróci błąd. Jest to konsekwencja

fakt, że odbiornik sprawdza, czy bity $k1$ młodszego rzędu \hat{m} mają wartość 0 deszyfrowanie; jeśli atakujący nie skonstruował jakiegoś zaszyfrowanego tekstu c , legalnie szyfrując jakąś wiadomość, prawdopodobieństwo spełnienia tego warunku jest znikome. Formalny dowód komplikuje fakt, że na zapytania wyroczni atakującego należy odpowiedzieć poprawnie, bez wiedzy prywatnego key, co oznacza, że musi istnieć skuteczny sposób określenia, czy zwrócić błąd, czy nie, a jeśli nie, jaki komunikat zwrócić. To zostaje osiągnięte patrząc na zapytania przeciwnika kierowane do przypadkowych wyroczni G, H.

Atak wybranego szyfrogramu Mangera na PKCS nr 1 v2.0. W 2001 roku Jakub Manger pokazał atak wybranym szyfrogramem na niektóre implementacje schematu szyfrowania RSA określony w PKCS #1 v2.0 —choć jaki określono, że był to wariant RSA-OAEP! Skoro Construction 11.36 jest bezpieczny pod względem CCA (zakładając, że problem RSA jest trudny), jak to możliwe?

Badając algorytm deszyfrowania w Construction 11.36, zwróć uwagę, że tam są dwa sposoby wystąpienia błędu: albo $\hat{m} \in Z_N$ jest za duży lub $m \in \{0, 1\}^{+k1}$ nie ma wystarczającej liczby końcowych zer. W konstrukcji 11.36 odbiorca powinien zwracać ten sam błąd (oznaczony σ) w obu przypadkach. Jednak w niektórych implementacjach odbiornik generowałby różne błędy w zależności od który krok się nie powiodł. Ten pojedynczy bit dodatkowych informacji umożliwia atakującemu przeprowadzić atak wybranym szyfrogramem, który odzyskuje wiadomość w całości z szyfrowania tej wiadomości, używając tylko około N zapytań do wyroczni który powoduje wyciek komunikatu o błędzie po odszyfrowaniu. To pokazuje, jak ważne jest wdrażanie schematów kryptograficznych dokładnie tak, jak określono, ponieważ wynikowy dowód i analiza mogą nie mieć już zastosowania w przypadku zmiany aspektów programu.

Nawet jeśli w obu przypadkach zostanie zwrócony ten sam błąd, osoba atakująca może określić, gdzie występuje błąd, jeśli czas zwrócenia błędu jest inny. (Ten to świetny przykład tego, jak osoba atakująca nie ogranicza się do sprawdzania wejść/wyjść algorytmu, ale może wykorzystać do ataku informacje z kanału bocznego schematu.) Wdrożenia muszą być ostrożne, aby zapewnić czas powrotu błąd jest identyczny niezależnie od tego, gdzie występuje błąd.

11.5.5 *KEM bezpieczny dla CCA w modelu Random-Oracle

Pokazujemy tutaj konstrukcję KEM opartą na RSA, która jest zabezpieczona CCA model losowej wyroczni. (Przypomnijmy sobie z Twierdzenia 11.14, że każda taka konstrukcja może być używana w połączeniu z dowolnym szyfrowaniem klucza prywatnego zabezpieczonym przez CCA schemat zapewniający schemat szyfrowania klucza publicznego zabezpieczony przez CCA.) W porównaniu do schematu RSA-OAEP z poprzedniej sekcji, główną zaletą jest prostota konstrukcji i dowód jej bezpieczeństwa. Jego główną wadą jest to, że podczas szyfrowania krótkich wiadomości powstają dłuższe teksty zaszyfrowane ponieważ wymaga paradygmatu KEM/DEM, podczas gdy RSA-OAEP nie. Dla szyfrowania długich wiadomości, jednak jako część będzie również używany RSA-OAEP hybrydowy schemat szyfrowania, co spowodowałoby, że schemat szyfrowania miałby wydajność podobną do tej, którą można uzyskać stosując pokazany tutaj KEM.

Opisywany przez nas KEM jest częścią normy ISO/IEC 18033-2 dotyczącej szyfrowania klucza publicznego. Na schemacie klucz publiczny zawiera jak zwykle N , e oraz określono funkcję $H : Z \rightarrow \{0, 1\}^n$, która w analizie będzie modelowana jako losowa wyrocznia. (Ta funkcja może opierać się na jakiejś kryptograficznej funkcji skrótu, jak omówiono w podrozdziale 5.5. Pomijamy szczegóły.) Aby enkapsułować klucz, nadawca wybiera uniform $r \in Z$, a następnie oblicza zaszyfrowany tekst $c := [r \text{ mod } N]$ i klawisz $k := H(r)$. Aby N odszyfrować tekst zaszyfrowany c , odbiorca po prostu odzyskuje r w zwykły sposób, a następnie ponownie uzyskuje ten sam klucz $k := H(r)$. Zobacz Konstrukcja 11.37.

BUDOWA 11.37

Niech GenRSA będzie jak zwykle i skonstruuje KEM w następujący

sposób:

- Gen: na wejściu $1n$, uruchom GenRSA($1n$), aby obliczyć (N, e, d) . The

wejściu $1n$ klucz publiczny to N , e , a klucz prywatny to N , d .

W ramach generowania klucza funkcja $H:Z$, ale $N \in \{0, 1\}^n$ podano n , pozostawiamy to ukryte.

- Encaps: na wejściowym kluczu publicznym N , $e \in 1n$, wybierz uniform $r \in \text{mod } Z \text{ mod } N$. Wypisz zaszyfrowany tekst $c := [r \text{ mod } N]$ i klucz $k :=$
- Decaps: na wejściu klucz prywatny N , d i tekst zaszyfrowany $c \in Z \text{ mod } N$ i n , oblicz $r := [c]^d \text{ mod } N$ na wyjściu klucz $k := H(r)$.

KEM bezpieczny dla CCA (w modelu random-oracle).

Bezpieczeństwo programu CPA jest natychmiastowe. Rzeczywiście, szyfrogram c jest równy $[r \text{ mod } N]$ dla jednolitego $r \in Z$, więc założenie RSA implikuje, że podsłuchiwacz, który zaobserwuje c , nie będzie w stanie obliczyć r . Oznacza to z kolei, że podsłuchujący nie będzie pytał r do H , a co za tym idzie o wartość klucza $k = H(r)$ pozostała jednolita z punktu widzenia atakującego.

W rzeczywistości powyższe dotyczy również bezpieczeństwa CCA. Dzieje się tak dlatego, że odpowiadanie na zapytanie wyroczni dekapsulacyjnej dotyczące dowolnego tekstu zaszyfrowanego $\tilde{c} = c$ tylko wymaga oceny H na pewnym wejściu $[\tilde{c} \text{ mod } N] = \tilde{r} = r$. Zatem zapytania wyroczni dekapsulacyjnej atakującego nie ujawniają żadnych dodatkowych informacji na temat klucza $H(r)$ zawartego w zaszyfrowanym tekście wyzwania. (Formalny dowód jest nieco bardziej skomplikowany, ponieważ musimy pokazać, jak możliwa jest symulacja odpowiedzi na zapytania wyroczni dekapsulacyjnej bez znajomości klucza prywatnego. Okazuje się jednak, że nie jest to zbyt trudne.)

TWIERDZENIE 11.38 Jeśli problem RSA jest trudny w porównaniu z GenRSA i H jest modelowany jako losowa wyrocznia, wówczas Konstrukcja 11.37 jest zabezpieczona CCA.

DOWÓD Niech Π oznacza Konstrukcję 11.37 i niech A będzie probabilistycznym przeciwnikiem wielomianowym w czasie. Dla wygody i ponieważ jest to pierwszy dowód, w którym wykorzystujemy pełną moc modelu losowej wyroczni, wyraźnie opisujemy etapy eksperymentu KEMcca A, $\Pi(n)$:

1. GenRSA(1^n) jest uruchamiane w celu otrzymania (N, e, d) . Dodatkowo wybierana jest $H: \mathbb{Z}_N$ funkcja losowa $\{0, 1\}^n$.
2. Obliczane są jednolite wybrany jest tekst zaszyfrowany $c := [r \text{ mod } N]$ i klucz $r \in \mathbb{Z}_N$ $k := H(r)$.
3. Wybierany jest bit jednolity $b \in \{0, 1\}$. Jeśli $b = 0$, ustaw $\hat{k} := k$. Jeżeli $b = 1$ to wybierz jednorodny $\hat{k} \in \{0, 1\}^n$.
4. A ma podane $pk = N, e, c$ i \hat{k} i może zapytać $H(\cdot)$ (na dowolnym wejściu) i wyrocznię dekapsulacyjną DecapsN,d (\cdot) na dowolnym zaszyfrowanym tekście $\hat{c} = c$.
5. A wyprowadza trochę b . Wynik eksperymentu definiuje się jako 1, jeśli $b = b$ i 0 w przeciwnym razie.

Podczas wykonywania eksperymentu KEMcca A, $\Pi(n)$ niech Query będzie zdarzeniem, które w dowolnym momencie jego wykonywania A zapyta r do losowej wyroczni H. Niech Sukces oznacza zdarzenie, że $b = b$ (tj. wyniki eksperymentu 1). Następnie

$$\Pr[\text{Sukces}] = \Pr[\text{Sukces} \mid \text{Zapytanie}] + \frac{\Pr[\text{Sukces} \mid \text{Zapytanie}]}{\Pr[\text{Sukces} \mid \text{Zapytanie}] + \Pr[\text{Zapytanie}]},$$

gdzie wszystkie prawdopodobieństwa przejmuje losowość zastosowana w doświadczeniu KEMcca A, $\Pi(n)$. Pokazujemy, że $\Pr[\text{Success} \mid \text{Query}]$ i $\Pr[\text{Query}]$ jest zaniedbywalny. Twierdzenie jest następujące.

Najpierw argumentujemy, że $\Pr[\text{Sukces} \mid \text{Zapytanie}] \geq \frac{1}{2}$. Jeżeli $\Pr[\text{Zapytanie}] = 0$, to jest im pośredniczy. W przeciwnym razie $\Pr[\text{Sukces} \mid \text{Zapytanie}] = \Pr[\text{Sukces} \mid \text{Zapytanie}]$. Teraz, uzależniona od Query, wartość prawidłowego klucza $k = H(r)$ jest jednolita, ponieważ H jest funkcją losową. Rozważmy informację A o k w doświadczeniu KEMcca A, $\Pi(n)$. Klucz publiczny pk i tekst zaszyfrowany c same w sobie nie zawierają żadnych informacji o k . (Określają jednoznacznie r , ale ponieważ H jest wybierane niezależnie od czegokolwiek innego, nie daje to żadnej informacji o $H(r)$.)

Zapytania, które A kieruje do H, również nie ujawniają żadnych informacji o r , chyba że A zadaje r do H (w takim przypadku następuje zapytanie); to znowu opiera się na fakcie, że H jest funkcją losową. Wreszcie zapytania, które A zadaje swojej wyroczni dekapsulującej, ujawniają jedynie $H(\tilde{r})$ dla $\tilde{r} = r$. Wynika to z faktu, że $\text{DecapsN}, d(\tilde{c}) = H(\tilde{r})$ gdzie $\tilde{r} = [\tilde{c} \text{ mod } N]$, ale $\tilde{c} = c$ implikuje $\tilde{r} = r$. Po raz kolejny to oraz fakt, że H jest funkcją losową, oznacza, że żadna informacja o $H(r)$ nie zostanie ujawniona, chyba że pojawi się Query.

Powyższe pokazuje, że dopóki nie wystąpi Query, wartość prawidłowego klucza k jest jednakowa, nawet biorąc pod uwagę pogląd A na klucz publiczny, tekst zaszyfrowany i

odpowiedzi na wszystkie zapytania wyroczni. W takim przypadku A nie ma możliwości rozróżnienia (lepszego niż przypadkowe zgadywanie), czy \hat{k} jest właściwym kluczem, czy jednolitym, niezależnym kluczem. Dlatego Pr Sukces [Zapytanie] = $\frac{1}{2}$.

Podkreślamy, że w żadnym miejscu powyżej argumentacji nie opieraliśmy się na fakcie, że A jest ograniczone obliczeniowo i w rzeczywistości Pr Sukces [Zapytanie] nawet jeśli na A nie nałożono żadnych ograniczeń obliczeniowych. Wskazuje to na część mocy modelu losowej wyroczni.

Aby zakończyć dowód twierdzenia, pokażemy

Twierdzenie 11.39 Jeśli problem RSA jest trudny w porównaniu z GenRSA , a H jest modelowany jako losowa wyrocznia, wówczas Pr[Query] jest nieistotne.

Aby to udowodnić, konstruujemy algorytm A, który wykorzystuje A jako podprogram. A otrzymuje instancję N, e, c problemu RSA i jego celem jest obliczenie r, dla którego $r = c \bmod N$. Aby to zrobić, uruchom A, odpowiadając na jego zapytania do H i Decaps. Obsługa zapytań do H jest prosta, ponieważ A może po prostu zwrócić losową wartość. Zapytania do Decaps są jednak bardziej skomplikowane, ponieważ A nie zna klucza prywatnego powiązanego z efektywnym kluczem publicznym N, np.

Jednak po głębszym zastanowieniu odpowiedzi na zapytania dotyczące dekapsulacji również są łatwe, ponieważ A może również tutaj zwrócić losową wartość. Oznacza to, że chociaż zapytanie Decaps(\tilde{c}) powinno zostać obliczone najpierw przez obliczenie \tilde{r} w taki sposób, że $\tilde{r} = \tilde{c} \bmod N$, a następnie obliczenie $H(\tilde{r})$, wynikiem jest po prostu jednolita wartość. Zatem A może po prostu zwrócić losową wartość bez wykonywania obliczeń pośrednich. Jedynym „haczykiem” jest to, że A musi zapewnić spójność pomiędzy swoimi odpowiedziami na zapytania H i zapytania Decaps; mianowicie musi zapewnić, że $H(\tilde{r}) = \text{Decaps}(\tilde{c})$. Oznacza to, i list LH i LDecaps, które śledzą, że dla dowolnego $\tilde{r}, \tilde{c} \sim r$ obsługiwane przy użyciu prostej księgowości odpowiedzi, których A udzielił w odpowiedzi na odpowiednie zapytania Oracle. Teraz podajemy szczegóły.

Algorytm A: Jako dane wejściowe podaje się algorytm (N, e, c).

1. Zainicjuj puste listy LH, LDecaps. Wybierz uniform $k \in \{0, 1\}^n$ i zapisz (c, k) w LDecaps.

2. Wybierz bit jednolity $b \in \{0, 1\}$. Jeśli $b = 0$, ustaw $\hat{k} := k$. Jeżeli $b = 1$ to wybierz uniform $\hat{k} \in \{0, 1\}^n$. Uruchom A na N, e, c i \hat{k} .

Kiedy A zadaje pytanie $H(\tilde{r})$, odpowiedz na nie w następujący sposób:

- Jeżeli w LR występuje zapis w postaci (\tilde{r}, k) dla jakiegoś k , zwróć k .
- W przeciwnym razie niech $\tilde{c} := [\tilde{r} \bmod N]$. Jeżeli w LDecaps znajduje się wpis w postaci (\tilde{c}, k) dla jakiegoś k , zwróć k i zapisz (\tilde{r}, k) w LH.

- W przeciwnym razie wybierz uniform $k \in \{0, 1\}^n$, zwróć k i zapisz (\tilde{r}, k) w LR.

Kiedy A zadaje pytanie Decaps(\tilde{c}), odpowiedz na nie w następujący sposób:

- Jeśli w LDecaps znajduje się wpis w postaci (\tilde{c}, k) dla jakiegoś k , zwróć k .
- W przeciwnym razie dla każdego wpisu $(\tilde{r}, k) \in LH$ sprawdź, czy $\tilde{r}^{m_i} = c \mod N$ i, jeśli tak, wyrowadź k .
- W przeciwnym razie wybierz uniform $k \in \{0, 1\}^n$, zwróć k i zapisz (\tilde{c}, k) w LDecaps.

3. Na końcu wykonywania A, jeśli istnieje wpis (r, k) w $LH = c \mod N$, dla którego $r^{m_i} \equiv k \pmod{N}$

Jest oczywiste, że A działa w czasie wielomianowym, a widok A uruchamiany jako podprogram przez A w eksperymencie RSA-invA, GenRSA(n) jest identyczny z widokiem A w eksperymencie KEMcca A, $\Pi(n)$: dane wejściowe podane A mają wyraźnie właściwy rozkład, odpowiedzi na zapytania wyroczni A są spójne, a odpowiedzi na wszystkie zapytania H są jednolite i niezależne. Na koniec A podaje prawidłowe rozwiązanie dokładnie w momencie wystąpienia zapytania. Twardość problemu RSA w stosunku do GenRSA oznacza zatem, że $\Pr[Query]$ jest nieistotne, zgodnie z wymaganiami. ■

Warto zwrócić uwagę na różne właściwości modelu losowej wyroczni (patrz podrozdział 5.5.1), które zostały użyte w powyższym dowodzie. Po pierwsze, opieramy się na fakcie, że wartość $H(r)$ jest jednolita, chyba że r jest odpytywane do H —nawet jeśli H jest odpytywane o wiele innych wartości $\tilde{r} = r$. W sposób dorozumiany używamy możliwości wyodrębnienia, aby argumentować, że atakujący nie może wysłać zapytania r do H ; w przeciwnym razie moglibyśmy wykorzystać tego atakującego do rozwiązania problemu RSA. Wreszcie dowód opiera się na programowalności w celu symulacji zapytań przeciwnika o dekapsulację-wyrocznię.

11.5.6 Problemy i pułapki związane z wdrażaniem RSA

Zamykamy tę sekcję krótkim omówieniem niektórych zagadnień związanych z wdrażaniem schematów opartych na RSA i pewnych pułapek, o których należy pamiętać.

Korzystanie z chińskiej pozostałości. W implementacjach szyfrowania opartego na RSA odbiorca może użyć chińskiego twierdzenia o resztach (sekcja 8.1.5), aby przyspieszyć obliczanie pierwiastków eth modulo N podczas deszyfrowania. W szczególności niech $N = pq$ i powiedzmy, że odbiorca chce obliczyć pierwiastek eth z pewnej wartości y , używając $d = [e \text{ mod } \varphi(N)]$. Odbiorca może wykorzystać korespondencję $[y \text{ mod } p], [y \text{ mod } q]$ do obliczenia wyników częściowych

$$xp := [y^{-D} p \text{ mod } p] = y[d \text{ mod } (p-1)] \text{ mod } p \quad (11.19)$$

I

$$xq := [y^{-D} q, \text{ mod } q] = y[d \text{ mod } (q-1)] \text{ mod } q \quad (11.20)$$

a następnie połącz je, aby otrzymać $x = (xp, xq)$, jak omówiono w Sekcji 8.1.5.

Zauważ, że $[d \bmod (p-1)]$ i $[d \bmod (q-1)]$ można obliczyć wcześniej, ponieważ są one niezależne od y .

Dlaczego to jest lepsze? Założymy, że potęgowanie modulo -bitowej liczby całkowitej wykonuje $y \cdot 3^3$ operacje dla pewnej stałej y . Jeśli p, q mają długość n bitów, wówczas naiwnie wykonuje obliczenie $[y \bmod N]$ zajmuje $y \cdot (2n) = 8y \cdot n$ Użycie reszty 3^3 się kroki (ponieważ $N = 2n$).) kroki chińskiej zmniejsza to do w przybliżeniu $2 \cdot (y \cdot np = q = n)$, czyli w 3^3 (ponieważ przybliżeniu 1/4 czasu.

Przykład 11.40

Powracamy do przykładu 8.49. Przypomnijmy, że $N = 143 = 11 \cdot 13$ i $d = 103$, a tam $y = 64$. Aby obliczyć $[64103 \bmod 143]$, wykonujemy obliczenia

$$\begin{aligned}[64 \bmod 11], [64 \bmod 13]103 &= [(-2)103 \bmod 11], [(-1)103 \bmod 13] \\ &= [(-2)[103 \bmod 10] \bmod 11], 1 \\ &= [-8 \bmod 11], 1 = (3, 1).\end{aligned}$$

Możemy obliczyć $1p = 78 \quad (1, 0)$ i $1q = 66 \quad (0, 1)$, jak omówiono w podrozdziale 8.1.5.

(Zauważ, że te wartości można wstępnie obliczyć, ponieważ są niezależne od y .) Wtedy $(3, 1)$

$3 \cdot 1p + 1q = 3 \cdot 78 + 66 = 168 = 25 \bmod 143$, w zgodność z odpowiedzią uzyskany wcześniej.

Atak błędu podczas używania chińskiej reszty. Używając chińskiej reszty zgodnie z opisem, należy mieć świadomość potencjalnego ataku, który może zostać przeprowadzony w przypadku wystąpienia błędów (lub może zostać wywołany przez osobę atakującą, np. w wyniku manipulacji sprzętowej) w trakcie obliczeń.

Zastanów się, co się stanie, jeśli $[y^D \bmod N]$ jest obliczany dwukrotnie: za pierwszym razem bez błędu (dając poprawny wynik x), ale za drugim razem z błędem podczas obliczania równania (11.20), ale nie równania (11.19) (ten sam atak stosuje się w odwrotnym przypadku). Drugie obliczenie daje niepoprawny wynik x , dla którego $x = x \bmod p$, ale $x = x \bmod q$. Oznacza to, że $p \mid (x - x)$ ale $q \mid (x - x)$. Ale wtedy $\gcd(x - x, N) = p$, co daje rozkład na czynniki N .

Jednym z możliwych środków zaradczych jest sprawdzenie poprawności wyniku przed jego użyciem, poprzez sprawdzenie, czy $x \bmod N$ nadal daje lepszą wydajność. Jest to zalecane w implementacjach sprzętowych.

Zależne klucze publiczne I. Jeżeli wielu odbiorców chce używać tego samego schematu szyfrowania, powinni używać niezależnych kluczy publicznych. Ten i następny atak pokazują, co może pójść nie tak, jeśli nie zostanie to zrobione.

Wyobraź sobie, że firma chce zastosować ten sam moduł N dla każdego ze swoich pracowników. Ponieważ nie jest pożądane, aby wiadomości zaszyfrowane dla jednego pracownika były czytane przez innego pracownika, firma udostępnia różne pary (e_i, d_i)

każdego pracownika. Oznacza to, że klucz publiczny i-tego pracownika to $pki = N, ei$, a jego klucz prywatny to $sk = N, di$, gdzie $ei \cdot di = 1 \bmod \varphi(N)$ dla wszystkich i .

Takie podejście jest niepewne i pozwala każdemu pracownikowi czytać wiadomości zaszyfrowane dla wszystkich pozostałych pracowników. Powodem jest to, że jak zauważono w podrozdziale 8.2.4, obliczyć ei . Biorąc di z $ei \cdot di = 1 \bmod \varphi(N)$, można podać rozkład na czynniki N i efektywnie pod uwagę rozkład N na czynniki, możliwe jest modyfikację $\varphi(N)$ dla dowolnego j . compute $dj := e$

II. Pokazany właśnie 1 Zależne klucze publiczne

atak umożliwia dowolnemu pracownikowi odszyfrowanie wiadomości wysyłanych do dowolnego innego pracownika. To wciąż pozostawia możliwość, że dzielenie się modelem N będzie w porządku, o ile wszyscy pracownicy ufają sobie nawzajem (lub, alternatywnie, o ile należy zachować poufnosć jedynie przed osobami z zewnątrz, ale nie przed innymi członkami firmy). Tutaj pokazujemy scenariusz wskazujący, że udostępnianie modułu jest nadal złym pomysłem, przynajmniej w przypadku stosowania zwykłego szyfrowania RSA.

Załóżmy, że ta sama wiadomość m jest szyfrowana i wysyłana do dwóch różnych (znanych) pracowników za pomocą kluczy publicznych $(N, e1)$ i $(N, e2)$, gdzie $e1 = e2$. Założymy dalej, że $\gcd(e1, e2) = 1$. Następnie podsłuchujący widzi dwa szyfrogramy

$$c1 = me1 \bmod N \text{ i } c2 = me2 \bmod N.$$

Ponieważ $\gcd(e1, e2) = 1$, istnieją liczby całkowite X, Y takie, że $Xe1 + Ye2 = 1$ zgodnie z Twierdzeniem 8.2. Ponadto, mając publiczne wykładniki $e1$ i $e2$, możliwe jest efektywne obliczenie X i Y przy użyciu rozszerzonego algorytmu Euklidesa (patrz Załącznik B.1.2). Twierdzimy, że $m = [c \bmod N]$, co można łatwo obliczyć. Jest to prawdą, ponieważ
$$\begin{matrix} X \\ 1 \end{matrix} \cdot \begin{matrix} Y \\ e2 \end{matrix}$$

$$\begin{matrix} X \\ \text{do } 1 \end{matrix} \cdot \begin{matrix} Y \\ \text{do } 2 \end{matrix} = mXe1 + mYe2 = mXe1 + Y e2 = m1 = m \bmod N.$$

Podobny atak ma zastosowanie w przypadku użycia dopełnianego RSA lub RSA-OAEP, jeśli nadawca używa tej samej przekształconej wiadomości podczas szyfrowania do dwóch użytkowników.

Jakość losowości w generowaniu klucza RSA. W całej księdze zawsze zakładamy, że uczciwe strony mają dostęp do wystarczającej losowości wysokiej jakości. Kiedy to założenie zostanie naruszone, bezpieczeństwo może nie zostać spełnione.

W szczególności, jeśli ciąg -bitowy zostanie wybrany z jakiegoś zestawu $S = \{0, 1\}$, a nie równomiernie z $\{0, 1\}$, wówczas osoba atakująca może przeprowadzić wyszukiwanie metodą brute-force (w czasie $O(|S|)$) do ataku na system.

W niektórych przypadkach sytuacja może być jeszcze gorsza. Rozważmy w szczególności przypadek generowania klucza RSA, gdzie jedna próbka losowych bitów rp jest używana do wybrania pierwszej liczby pierwszej p , a druga próbka rq jest używana do generowania drugiej liczby pierwszej q . Założymy dalej, że wiele kluczy publicznych/prywatnych jest generowanych przy użyciu tego samego źródła losowości niskiej jakości, w którym rp, rq są wybierane równomiernie z pewnego zbioru S o rozmiarze $2s$. Po wygenerowaniu około $2s/2$ kluczy publicznych (patrz Dodatek A.4) spodziewamy się otrzymać dwa różne moduły N, N , które zostały wygenerowane przy użyciu identycznego współczynnika losowości $rp = r$, który można łatwo znaleźć p . Te dwa moduły mają wspólny a obliczając $\gcd(N, N)$. Napastnik

może w ten sposób przeszukać Internet w poszukiwaniu dużego zestawu kluczy publicznych RSA, obliczyć ich gcd w parach i mieć nadzieję na uwzględnienie jakiegoś ich podzbioru. Chociaż obliczanie parami gcd o modułach $2s/2$ naiwnie wymagałoby czasu $O(2s)$, okazuje się, że można to znacznie poprawić, stosując podejście „dziel i zwyciężaj”, które wykracza poza zakres tej książki. W rezultacie osoba atakująca może przeprowadzić wyszukiwanie metodą brute-force w czasie znacznie krótszym niż 2 sekundy. Co więcej, atak działa nawet wtedy, gdy atakujący nie zna zestawu S!

Powyższy scenariusz został zweryfikowany eksperymentalnie przez dwa niezależne pracujące zespoły badawcze, które przeprowadziły dokładnie powyższy atak na klucze publiczne zeskanowane w Internecie i udało im się pomyślnie rozłożyć na czynniki znaczną część znalezionych kluczy.

Referencje i dodatkowe lektury

Pomysł szyfrowania kluczem publicznym został po raz pierwszy zaproponowany w otwartej literaturze przez Diffiego i Hellmana [58]. Rivest, Shamir i Adleman [148] wprowadzili założenie RSA i zaproponowali schemat szyfrowania klucza publicznego oparty na tym założeniu. Jak wskazano w poprzednim rozdziale, do innych pionierów kryptografii klucza publicznego należą Merkle i Rabin (w publikacjach akademickich) oraz Ellis, Cocks i Williamson (w publikacjach tajnych).

Definicja 11.2 ma swoje korzenie w przełomowych pracach Goldwassera i Micali [80], którzy również jako pierwi uznali konieczność szyfrowania probabilistycznego w celu spełnienia tej definicji. Jak zauważono w rozdziale 4, ataki z użyciem wybranego tekstu zaszyfrowanego zostały po raz pierwszy formalnie zdefiniowane przez Naora i Yunga [129] oraz Rackoffa i Simona [147]. Artykuł objaśniający Shoupa [156] omawia znaczenie bezpieczeństwa przed atakami z użyciem wybranego tekstu zaszyfrowanego. Bellare i in. dają ujednolicone, nowoczesne podejście do różnych koncepcji bezpieczeństwa szyfrowania klucza publicznego.

Dowód bezpieczeństwa CPA dla szyfrowania hybrydowego został po raz pierwszy przedstawiony przez Bluma i Goldwassera [36]. Sprawę bezpieczeństwa CCA omówiono w [56].

Co nieco zdumiewające, schemat szyfrowania El Gamala [70] został zaproponowany dopiero w 1984 r., mimo że można go postrzegać jako bezpośrednią transformację protokołu wymiany kluczy Diffiego–Hellmana (patrz ćwiczenie 11.4). DHIES został wprowadzony w [2]. Normę ISO/IEC 18033-2 dotyczącą szyfrowania klucza publicznego można znaleźć pod adresem <http://www.shoup.net/iso>.

Zwykłe szyfrowanie RSA odpowiada oryginalnemu schematowi wprowadzonemu przez Rivesta, Shamira i Adlemana [148]. Ataki na zwykłe szyfrowanie RSA opisane w sekcji 11.5.1 wynikają z [161, 55, 84, 47, 40]; zobacz [120, rozdział 8] i [38], aby uzyskać dodatkowe ataki i dalsze informacje. Dowody twierdzenia Copper-smitha można znaleźć w oryginalnej pracy [46] lub kilku kolejnych objaśnieniach (np. [119]).

Standardy kryptograficzne RSA PKCS #1 (zarówno poprzednie, jak i obecne

wersje) są dostępne pod adresem <http://www.emc.com/emc-plus/rsa-labs>. Opisany tutaj atak wybranym tekstem jawnym na PKCS #1 v1.5 wynika z [49]. Opis ataku Bleichenbacher wybranym szyfrogramem na PKCS #1 v1.5 można znaleźć w oryginalnej pracy [34]. Dalsze ulepszenia można znaleźć w [12].

Dowody Twierdzenia 11.31 i uogólnienia można znaleźć w [8, 86, 66, 7].

Ogólne omówienie schematów tej formy można znaleźć w sekcji 13.1.2. Wydaje się, że konstrukcja 11.37 została wprowadzona i po raz pierwszy przeanalizowana przez Shoupa [157].

OAEP wprowadzili Bellare i Rogaway [22]. Później uznano, że oryginalny dowód OAEP jest błędny; odsyłamy zainteresowanego czytelnika [39, 158, 68]. Aby poznać szczegóły ataku Mangera przy użyciu wybranego tekstu zaszyfrowanego na implementacje PKCS #1 v2.0, zobacz [117].

Atak parami-gcd opisany w sekcji 11.5.6 został przeprowadzony przez Lenstra i in. [112] oraz Heninger i in. [88].

Stosując w praktyce dowolny schemat szyfrowania pojawia się pytanie, jakiej długości klucza użyć. Nie należy lekceważyć tej kwestii, dlatego odsyłamy czytelnika do sekcji 9.3 i zawartych w niej odniesień w celu uzyskania szczegółowego omówienia.

Pierwszy skuteczny schemat szyfrowania klucza publicznego z zabezpieczeniem CCA, nieopierający się na modelu random-oracle, pokazali Cramer i Shoup [50] w oparciu o założenie DDH. Następnie Hoffheinz i Kiltz pokazali skuteczny schemat bezpieczny dla CCA bez przypadkowych wyroczni, oparty na założeniu RSA [92].

Ćwiczenia

11.1 Załącz schemat szyfrowania kluczem publicznym dla wiadomości jednobitowych bez błędu deszyfrowania. Pokaż, że mając pk i szyfrogram c obliczony za pomocą $c = Enc_{pk}(m)$, nieograniczony przeciwnik może określić m z prawdopodobieństwem 1.

11.2 Pokaż, że w przypadku dowolnego schematu szyfrowania kluczem publicznym zabezpieczonego CPA dla wiadomości jednobitowych długość tekstu zaszyfrowanego musi być superlogarytmiczna w parametrze bezpieczeństwa.

Wskazówka: Jeśli nie, zakres możliwych szyfrogramów ma rozmiar wielomianowy.

11.3 Załączmy, że schemat szyfrowania kluczem publicznym (Gen , Enc , Dec) dla wiadomości n-bitowych jest jednokierunkowy, jeśli jakikolwiek przeciwnik $ppt A$ ma znikome prawdopodobieństwo powodzenia w następującym eksperymentie:

- $Gen(1^n)$ jest uruchamiany w celu uzyskania kluczy (pk, sk) .
- Wiadomość $m \in \{0, 1\}^n$ jest wybierana równomiernie i losowo i obliczany jest szyfrogram $c = Enc_{pk}(m)$.
- A otrzymuje pk i c i wysyła komunikat m . Mówimy A powiedzie się, jeśli $m = m$.

- (a) Podaj konstrukcję KEM z zabezpieczeniem CPA w modelu losowej Oracle w oparciu o dowolny jednokierunkowy schemat szyfrowania kluczem publicznym.
- (b) Czy deterministyczny schemat szyfrowania klucza publicznego może być jednokierunkowy? Jeśli nie, udowodnij niemożność; jeśli tak, podaj konstrukcję opartą na którymkolwiek z założeń przedstawionych w tej książce.

11.4 Pokaż, że dowolny protokół dwurundowej wymiany kluczy (to znaczy, w którym każda ze stron wysyła pojedynczą wiadomość) spełniający Definicję 10.1, można przekształcić w schemat szyfrowania kluczem publicznym bezpiecznym według CPA.

11.5 Wykazać, że stwierdzenie 11.7 nie ma zastosowania w przypadku zabezpieczeń CCA.

11.6 Rozważ następujący schemat szyfrowania kluczem publicznym. Klucz publiczny to (G, q, g, h) , a klucz prywatny to x , wygenerowany dokładnie tak, jak w schemacie szyfrowania El Gamala. Aby zaszyfrować bit b , nadawca wykonuje następujące czynności:

- (a) Jeśli $b = 0$, to wybierz uniform $y \in Z_q$ i oblicz $c_1 := g^y$, $c_2 := h^y$.
Zaszyfrowany tekst to c_1, c_2 .
- (b) Jeśli $b = 1$, to wybierz niezależny uniform $y, z \in Z_q$, oblicz y^z i $c_2 := g^{y^z}$, i ustaw szyfrrogram równy c_1, c_2 .

Pokaż, że możliwe jest efektywne odszyfrowanie danej wiedzy o x .

Udowodnić, że ten schemat szyfrowania jest bezpieczny pod względem CPA, jeśli decyzyjny problem Diffiego–Hellmana jest trudny w porównaniu z problemem G.

11.7 Rozważmy następujący wariant szyfrowania El Gamala. Niech $p = 2q + 1$, niech G będzie grupą kwadratów modulo p (więc G jest podgrupą Z_p rzędu q), i niech g będzie generatorem G . Klucz prywatny to (G, g, q, x) a klucz publiczny to (G, g, q, h) , gdzie $h = g^{x+1} \in Z_p$. Z_q jest dobierane jednolicie. Aby zaszyfrować wiadomość $m \in Z_q$, wybierz mundur $r \in Z_q$, oblicz $c_1 := g^{r \text{ mod } p}$, $c_2 := h^{r \text{ mod } p} + m \text{ mod } p$ i niech szyfrrogramem będzie c_1, c_2 . Czy ten program jest bezpieczny pod względem CPA? Udowodnij swoją odpowiedź.

11.8 Rozważmy następujący protokół dla dwóch stron A i B w celu rzucenia uczciwej monety (bardziej skomplikowane wersje tego protokołu można zastosować w przypadku gier hazardowych w Internecie): (1) zaufana strona T publikuje swój klucz publiczny pk ; (2) następnie A wybiera jednolity bit b_A , szyfruje go za pomocą pk i ogłasza tekst zaszyfrowany c_A i T ; (3) następnie B działa symetrycznie i ogłasza zaszyfrowany tekst $c_B = c_A$; (4) T odszyfrowuje zarówno c_A , jak i c_B , a strony XOR wyników w celu uzyskania wartości monety.

- (a) Argumentuj, że nawet jeśli A jest nieuczciwy (ale B jest uczciwy), to ostateczna wartość monety jest równomiernie rozłożony.
- (b) Założmy, że strony używają szyfrowania El Gamal (gdzie bit b znajduje się przed zakodowany jako element grupy g , c_B zaszyfrowaniem – zwróć na to uwagę). Skuteczne odszyfrowanie jest nadal możliwe). Pokaż, jak nieuczciwy B może ustawić monetę na dowolną wartość.

- (c) Zaproponuj, jaki rodzaj schematu szyfrowania byłby odpowiedni do zastosowania w tym miejscu.
 Czy możesz zdefiniować odpowiednie pojęcie bezpieczeństwa i udowodnić, że Twoja sugestia spełnia tę definicję?

11.9 Udowodnij formalnie, że schemat szyfrowania El Gamal nie jest zabezpieczony przez CCA.

- 11.10 W sekcji 11.4.4 pokazaliśmy, że szyfrowanie El Gamala jest plastyczne, a konkretnie, że mając zaszyfrowany tekst c_1, c_2 , który jest szyfrowaniem jakiejś nieznanej wiadomości m , możliwe jest wygenerowanie tekstu zaszyfrowanego c_1, c_2 , który jest szyfrowaniem $a \cdot m$ (dla znanego a). Odbiorca, który otrzyma oba te zaszyfrowane teksty, może być podejrzliwy, ponieważ oba zaszyfrowane teksty mają ten sam pierwszy składnik. Pokaż, że możliwe jest wygenerowanie c , czyli szyfrowania $a \cdot m$, gdzie $c = c_1$ i c

ok. 1 , 2

$$1 = c_2, 2$$

11.11 Dowód twierdzenia 11.22.

- 11.12 Jeden z ataków na zwykły RSA omówiony w sekcji 11.5.1 dotyczy nadawcy, który szyfruje dwie powiązane wiadomości przy użyciu tego samego klucza publicznego.
 Sformułuj odpowiednią definicję bezpieczeństwa wykluczającą takie ataki i pokaż, że dowolny schemat szyfrowania klucza publicznego z zabezpieczeniem CPA spełnia Twoją definicję.

- 11.13 Jeden z ataków na zwykły RSA omówiony w sekcji 11.5.1 dotyczy nadawcy, który szyfruje tę samą wiadomość do trzech różnych odbiorców.
 Sformułuj odpowiednią definicję bezpieczeństwa wykluczającą takie ataki i pokaż, że dowolny schemat szyfrowania klucza publicznego z zabezpieczeniem CPA spełnia Twoją definicję.

- 11.14 Rozważmy następującą zmodyfikowaną wersję szyfrowania dopełnianego RSA: Przyjmijmy, że wiadomości, które mają być zaszyfrowane, mają długość dokładnie $N/2$. Aby zaszyfrować, najpierw oblicz $\hat{m} := 0x00r0x0m$ gdzie r jest jednolitym ciągiem o długości $N/2 - 16$. Następnie oblicz zaszyfrowany tekst $c := [\hat{m} \text{ mod } N]$. Podczas deszyfrowania tekstu zaszyfrowanego c odbiorca oblicza $\hat{m} := [c \text{ mod } N]$ i zwraca błąd, jeśli \hat{m} nie składa się z $0x00$, po którym następuje $N/2 - 16$ dowolnych bitów, po których następuje $0x00$. Pokaż, że ten schemat nie jest bezpieczny dla CCA. Dlaczego łatwiej jest skonstruować atak wybranym szyfrogramem w tym schemacie niż w PKCS #1 v1.5?

- 11.15 Rozważmy schemat szyfrowania oparty na RSA, w którym użytkownik szyfruje wiadomość $m \in \{0, 1\}$ w odniesieniu do klucza publicznego N, e , obliczając $\hat{m} := H(m)m$ i wysyłając zaszyfrowany tekst $[\hat{m} \text{ mod } N]$. (Tutaj niech $H : \{0, 1\} \rightarrow \{0, 1\}$ n i założmy $n < N$.) Odbiornik odzyskuje \hat{m} w zwykły sposób i sprawdza, czy ma ono poprawną postać przed wyświetleniem najmniej znaczących bitów jako M . Udowodnij lub obal, że ten schemat jest bezpieczny pod względem CCA, jeśli H jest modelowany jako losowa wyrocznia.

11.16 Pokaż atak wybranym szyfrogramem na Construction 11.34.

11.17 Niech $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ będzie schematem szyfrowania klucza publicznego z zabezpieczeniem CPA, i niech $\Pi = (\text{Gen}, \text{schemat}, \text{Dec})$ będzie szyfrowaniem klucza prywatnego z zabezpieczeniem CCA Enc . Rozważmy następującą konstrukcję:

BUDOWA 11.41

Niech $H : \{0, 1\}^n \rightarrow \{0, 1\}^n$ będzie funkcją. Skonstruuj schemat szyfrowania klucza publicznego w następujący sposób:

- Gen : na wejściu $1n$, uruchom $\text{Gen}(1n)$, aby uzyskać (pk, sk) . Wyjście są to odpowiednio klucz publiczny i prywatny.
- Enc : po wprowadzeniu klucza publicznego pk i wiadomości $m \in \{0, 1\}^n$, wybieramy uniform $r \in \{0, 1\}^n$ i wysyłamy zaszyfrowany tekst $\text{Enc}_{pk}(r), \text{Enc}_H(r)(m)$.
- Dec : po wprowadzeniu klucza prywatnego sk i tekstu zaszyfrowanego c_1, c_2 , oblicz $r := \text{Dec}_{sk}(c_1)$ i ustaw $k := H(r)$. Następnie wyrowadź $\text{Dec}_k(c_2)$.

Czy powyższa konstrukcja ma nieroróżnialne szyfrowanie w przypadku ataku wybranym szyfrogramem, jeśli H jest modelowany jako losowa wyrocznia? Jeśli tak, przedstaw dowód. Jeśli nie, gdzie zawodzi podejście zastosowane do udowodnienia Twierdzenia 11.38?

11.18 Rozważmy następujący wariant konstrukcji 11.32:

BUDOWA 11.42

Niech GenRSA będzie jak zwykle i zdefiniuje schemat szyfrowania klucza publicznego w następujący sposób:

- Gen: na wejściu $1n$, uruchom $\text{GenRSA}(1n)$, aby otrzymać (N, e, d) . Wyprowadź klucz publiczny $pk = N$, e i klucz prywatny $sk = N, d$.
- Enc: na wejściu klucz publiczny $pk = N, e$ i wiadomość N . Wyprowadź $m \in \{0, 1\}$, wybierz uniform $r \in Z[r \in \text{mod } N]$, $\text{lsb}(r) = m$.
- Dec: na wejściu klucz prywatny $sk = N, d$ i szyfrogram c, b , oblicz $r := [c^{d \text{ mod } N}] \text{ mod } N$ i wyjście $\text{lsb}(r) = b$.

Udowodnij, że ten schemat jest bezpieczny pod względem CPA. Omów zalety i wady w porównaniu z wersją Construction 11.32.

11.19 Założymy, że trzech użytkowników ma klucze publiczne RSA $N_1, 3, N_2, 3$ i $N_3, 3$ (tj. wszyscy używają $e = 3$), przy czym $N_1 < N_2 < N_3$. Rozważ następującą metodę wysłania tej samej wiadomości $m \in \{0, 1\}$ do każdej z tych stron: wybierz mundur $r \in Z[N_1, N_3]$, i wyślij każdemu ten sam szyfrogram

$$[r^3 \text{ mod } N_1], [r^3 \text{ mod } N_2], [r^3 \text{ mod } N_3], H(r) \cdot m ,$$

gdzie $H : Z[N_1, N_3] \rightarrow \{0, 1\}$. Przypuszczać N_1, N_2, N_3 .

- (a) Pokaż, że nie jest to bezpieczne CPA i przeciwnik może odzyskać m z zaszyfrowanego tekstu, nawet jeśli H jest modelowane jako losowa wyrocznia.

Wskazówka: zobacz rozdział 11.5.1.

- (b) Pokaż prosty sposób naprawienia tego problemu i uzyskaj metodę zabezpieczoną CPA, która przesyła tekst zaszyfrowany o długości $3 + O(n)$.

- (c) Pokaż lepsze podejście, które nadal jest bezpieczne CPA, ale z zaszyfrowanym tekstem o długości $+ O(n)$.

11.20 Napraw klucz publiczny RSA N , e i założmy, że mamy algorytm A , który zawsze poprawnie oblicza $\text{lsb}(x)$ przy danym $[x \text{ e mod } N]$. Napisz pełny pseudokod dla algorytmu A , który oblicza x z $[x \text{ e mod } N]$.

11.21 Napraw klucz publiczny RSA N , e i zdefiniuj

$$\text{połowa}(x) = \begin{cases} 0 & \text{jeśli } 0 < x < N/2 \\ 1 & \text{jeśli } N/2 < x < N \end{cases}$$

Udowodnić, że połowa jest twardym predykatem problemu RSA.

Wskazówka: zredukuj do twardości obliczania lsb .

Machine Translated by Google

Rozdział 12

Schematy podpisu cyfrowego

12.1 Podpisy cyfrowe – przegląd

W poprzednim rozdziale zbadaliśmy, w jaki sposób można wykorzystać szyfrowanie kluczem publicznym, aby zapewnić poufność ustawień klucza publicznego. Integralność (lub autentyczność) ustawień klucza publicznego jest zapewniana przy użyciu schematów podpisu cyfrowego. Można je postrzegać jako odpowiedniki klucza publicznego kodów uwierzytelniających wiadomości, chociaż, jak zobaczymy, istnieje kilka ważnych różnic między tymi kodami podstawowymi.

Schematy podpisu pozwalają podpisującemu S, który ustanowił klucz publiczny pk, na „podpisanie” wiadomości przy użyciu powiązanego klucza prywatnego sk w taki sposób, że każdy, kto zna pk (i wie, że ten klucz publiczny został ustanowiony przez S) może sprawdzić, czy wiadomość pochodzi od S i nie została zmodyfikowana podczas przesyłania. (Należy pamiętać, że w przeciwnieństwie do szyfrowania kluczem publicznym, w kontekście podpisów cyfrowych właściciel klucza publicznego pełni rolę nadawcy). Jako prototypową aplikację rozważmy firmę programistyczną, która chce rozpowszechniać aktualizacje oprogramowania w sposób uwierzytelniony; oznacza to, że gdy firma wypuści aktualizację, każdy z jej klientów powinien mieć możliwość sprawdzenia, czy aktualizacja jest autentyczna, a złośliwa strona trzecia nie powinna nigdy być w stanie oszukać klienta, aby zaakceptował aktualizację, która w rzeczywistości nie została wydana przez firmę firmę. Aby to zrobić, firma może wygenerować klucz publiczny pk wraz z kluczem prywatnym sk, a następnie dystrybuować pk w niezawodny sposób wśród swoich klientów, zachowując sk w tajemnicy. (Podobnie jak w przypadku szyfrowania kluczem publicznym, zakładamy, że ta początkowa dystrybucja klucza publicznego jest przeprowadzana prawidłowo, tak aby wszyscy klienci mieli poprawną kopię pk. W bieżącym przykładzie pk można było dołączyć do oryginalnego zakupionego oprogramowania przez klienta.) Wydając aktualizację oprogramowania m, firma oblicza podpis cyfrowy σ na m przy użyciu swojego klucza prywatnego sk i wysyła (m, σ) do każdego klienta. Każdy klient może zweryfikować autentyczność m sprawdzając, czy σ jest poprawnym podpisem na m w odniesieniu do klucza publicznego pk.

Złośliwa strona może spróbować wydać fałszywą aktualizację, wysyłając (m, σ) do klienta, gdzie m oznacza aktualizację, która nigdy nie została wydana przez firmę. Może to być zmodyfikowana wersja jakiejś poprzedniej aktualizacji lub może być zupełnie nowa i niepowiązana z żadnymi wcześniejszymi aktualizacjami. Jeśli jednak schemat podpisu jest „bezpieczny” (w pewnym sensie zdefiniujemy to wkrótce dokładniej), to gdy klient spróbuje zweryfikować σ, odkryje, że jest to nieprawidłowy podpis na m w odniesieniu do pk i dlatego odrzuci podpis. The

klient odrzuci, nawet jeśli m zostanie tylko nieznacznie zmodyfikowane w stosunku do oryginalnej aktualizacji m.

Powyższe nie jest tylko teoretycznym zastosowaniem podpisów cyfrowych, ale jednym który jest obecnie powszechnie używany do dystrybucji aktualizacji oprogramowania.

Porównanie z kodami uwierzytelniania wiadomości

Aby zapewnić integralność przesyłanych wiadomości, stosowane są zarówno kody uwierzytelniające wiadomości, jak i schematy podpisu cyfrowego. Chociaż dyskusja w Rozdziale 10 porównująca ustawienia klucza publicznego i prywatnego skupiała się głównie na szyfrowaniu, dyskusja ta dotyczy także integralności wiadomości. Używanie podpisów cyfrowych zamiast kodów uwierzytelniających wiadomości upraszcza dystrybucję kluczy i zarządzanie nimi, szczególnie gdy nadawca musi komunikować się z wieloma odbiorcami, jak w powyższym przykładzie aktualizacji oprogramowania. Stosując schemat podpisu cyfrowego, nadawca unika konieczności ustalania odrębnego tajnego klucza z każdym potencjalnym odbiorcą i unika konieczności obliczania osobnego znacznika MAC w odniesieniu do każdego takiego klucza. Zamiast tego nadawca musi obliczyć tylko jeden podpis, który może zostać zweryfikowany przez wszystkich odbiorców.

Jakościową zaletą podpisów cyfrowych w porównaniu z kodami uwierzytelniającymi wiadomości jest to, że podpisy można weryfikować publicznie. Oznacza to, że jeśli odbiorca zweryfikuje, czy podpis na danej wiadomości jest zgodny z prawem, wówczas wszystkie inne strony, które otrzymają tę podpisana wiadomość, również zweryfikują ją jako wiarygodną.

Tej funkcji nie można uzyskać za pomocą kodów uwierzytelniających wiadomości, jeśli podpisujący dzieli się oddzielnym kluczem z każdym odbiorcą: w takim ustawieniu złośliwy nadawca może obliczyć prawidłowy znacznik MAC w odniesieniu do klucza, który dzieli z odbiorcą A, ale nieprawidłowy znacznik MAC w odniesieniu do klucza, który dzieli z innym użytkownikiem B. W tym przypadku A wie, że otrzymał autentyczną wiadomość od nadawcy, ale nie ma gwarancji, że B się zgodzi.

Publiczna weryfikowalność oznacza, że podpisy można przenosić: podpis o na wiadomości m przez osobę podpisującą S może zostać pokazany osobie trzeciej, która może następnie sama zweryfikować, czy o jest prawidłowym podpisem na m w odniesieniu do klucza publicznego S (tutaj mamy założmy, że ta strona trzecia zna również klucz publiczny S). Tworząc kopię podpisu, ta osoba trzecia może następnie pokazać podpis innej stronie i przekonać ją, że S uwierzyteliła m, i tak dalej. Publiczna weryfikacja i możliwość przenoszenia są niezbędne do stosowania podpisów cyfrowych w certyfikatach i infrastrukturze klucza publicznego, co omówimy w sekcji 12.7.

Schematy podpisu cyfrowego zapewniają również bardzo ważną właściwość niezaprzecalności. Oznacza to, że gdy S podpisze wiadomość, nie może później zaprzeczyć, że to zrobił (zakładając, że klucz publiczny S jest szeroko nagłośniony i rozpowszechniany). Ten aspekt podpisów cyfrowych ma kluczowe znaczenie w zastosowaniach prawnych, w których odbiorca może być zmuszony do udowodnienia stronie trzeciej (powiedzmy sędziemu), że osoba podpisująca rzeczywiście „potwierdziła” określzoną wiadomość (np. umowę): zakładając, że osoba S klucz jest znany sędziemu lub jest w inny sposób publicznie dostępny, ważny podpis na wiadomości stanowi przekonujący dowód, że S rzeczywiście podpisał tę wiadomość. Kody uwierzytelniające wiadomości po prostu nie mogą zapewnić niezaprzecalności. Aby to zobaczyć, powiedzmy, że użytkownicy S i R mają wspólny klucz kSR, a S

wysyła wiadomość m do R wraz z (ważnym) znacznikiem MAC t obliczonym przy użyciu tego klucza. Ponieważ sędzia nie zna kSR (w rzeczywistości klucz ten jest utrzymywany w tajemnicy przez S i R), sędzia nie ma możliwości ustalenia, czy t jest ważne, czy nie. Gdyby R ujawnił sędziemu klucz kSR , sędzia nie miałby możliwości dowiedzenia się, czy jest to „prawdziwy” klucz, który udostępnili S i R, czy też jest to jakiś „fałszywy” klucz wyprodukowany przez R. Wreszcie , nawet jeśli założymy, że sędzia może w jakiś sposób uzyskać faktyczny klucz kSR wspólny dla stron, nie ma możliwości, aby sędzia rozróżnił, czy S wygenerował t, czy też R – wynika to z faktu, że kody uwierzytelniające wiadomości są kodami symetrycznymi klucz pierwotny; wszystko, co może zrobić S, może zrobić również R.

Podobnie jak w przypadku szyfrowania kluczem prywatnym i kluczem publicznym, kody uwierzytelniające wiadomości mają tę zaletę, że są krótsze i o około 2–3 rzędy wielkości skuteczniejsze w generowaniu/weryfikacji niż podpisy cyfrowe. Zatem w sytuacjach, gdy nie jest potrzebna publiczna weryfikowalność, przenosalność i/lub niezaprzecjalność, a nadawca komunikuje się przede wszystkim z jednym odbiorcą (z którym może udostępnić tajny klucz), należy zastosować kody uwierzytelniające wiadomość .

Związek z szyfrowaniem klucza publicznego

Podpisy cyfrowe są często błędnie postrzegane jako „odwrotność” szyfrowania kluczem publicznym, w której role nadawcy i odbiorcy są zamienione. Historycznie1 faktycznie sugerowano, że podpisy cyfrowe można uzyskać poprzez „odwrócenie” szyfrowania kluczem publicznym, tj. podpisanie wiadomości m poprzez jej odszyfrowanie (przy użyciu klucza prywatnego) w celu uzyskania σ i weryfikację podpisu σszyfrującego (za pomocą odpowiedniego klucza publicznego) i sprawdzając, czy wynikiem jest m. Sugestia, aby w ten sposób konstruować schematy podpisów, jest całkowicie bezpodstawną: w większości przypadków jest po prostu niemożliwa do zastosowania, a tam, gdzie ma zastosowanie, skutkuje to schematami podpisów, które nie są bezpieczne.

12.2 Definicje

Podpisy cyfrowe są odpowiednikiem kluczy publicznych kodów uwierzytelniających wiadomości, a ich składnia i gwarancje bezpieczeństwa są analogiczne. Algorytm, który nadawca stosuje do wiadomości, jest tutaj oznaczony jako Znak (a nie Mac), a wynik tego algorytmu jest teraz nazywany podpisem (a nie znacznikiem).

¹ Pogląd ten niewątpliwie powstał, ponieważ, jak zobaczymy w sekcji 12.4.1, zwykłe podpisy RSA są odwrotnością zwykłego szyfrowania RSA. Jednak ani zwykłe podpisy RSA, ani zwykłe szyfrowanie RSA nie spełniają nawet minimalnych założeń bezpieczeństwa.

Algorytm, który odbiorca stosuje do wiadomości i podpisu w celu sprawdzenia ważności, nadal jest oznaczony jako Vrfy.

DEFINICJA 12.1 Schemat podpisu (cyfrowego) składa się z trzech probabilistycznych algorytmów wielomianowych (Gen, Sign, Vrfy), takich jak:

1. Algorytm generowania klucza Gen przyjmuje jako dane wejściowe parametr bezpieczeństwa n i wyprowadza parę kluczy (pk, sk). Nazywa się je odpowiednio kluczem publicznym i kluczem prywatnym. Zakładamy, że pk i sk mają długość co najmniej n i że n można wyznaczyć na podstawie pk lub sk .
2. Algorytm podpisywania Sign przyjmuje jako dane wejściowe klucz prywatny sk i wiadomość m z jakiejś przestrzeni wiadomości (która może zależeć od pk). Wychodzi podpis σ i zapisujemy to jako $\sigma = \text{Sign}_{sk}(m)$.
3. Deterministyczny algorytm weryfikacji Vrfy przyjmuje jako dane wejściowe klucz publiczny pk , wiadomość m i podpis σ . Wyprowadza bit b , gdzie $b = 1$ oznacza prawidłowe, a $b = 0$ oznacza nieprawidłowe. Zapisujemy to jako $b := Vrfy_{pk}(m, \sigma)$.

Wymagane jest, aby poza znakomitym prawdopodobieństwem przekroczenia (pk, sk) wyniku Gen($1n$) utrzymywało się, że $Vrfy_{pk}(m, \text{Sign}_{sk}(m)) = 1$ dla każdego (legalnego) komunikatu m . Jeżeli istnieje funkcja taka, że dla każdego (pk, sk) wyjście Gen($1n$) przestrzeń komunikatów wynosi $\{0, 1\}^n$, to mówimy, że (Gen, Sign, Vrfy) jest schematem podpisu dla komunikatów długość (n).

Nazywamy σ prawidłowym podpisem wiadomości m (w odniesieniu do jakiegoś klucza publicznego pk rozumianego z kontekstu), jeśli $Vrfy_{pk}(m, \sigma) = 1$.

Schemat podpisu jest używany w następujący sposób. Jedna ze stron S , która pełni rolę nadawcy, uruchamia Gen($1n$) w celu uzyskania kluczy (pk, sk). Klucz publiczny pk jest następnie publikowany jako należący do S ; np. S może umieścić klucz publiczny na swojej stronie internetowej lub umieścić go w jakimś publicznym katalogu. Podobnie jak w przypadku szyfrowania kluczem publicznym, zakładamy, że każda inna strona jest w stanie uzyskać legalną kopię klucza publicznego S (patrz omówienie poniżej). Kiedy S chce uwierzytelnić wiadomość m , oblicza podpis $\sigma = \text{Sign}_{sk}(m)$ i wysyła (m, σ) . Po otrzymaniu (m, σ) odbiorca znający pk może zweryfikować autentyczność m , sprawdzając, czy $Vrfy_{pk}(m, \sigma) = 1$. Ustala to zarówno, że S wysłał m , jak i to, że

m nie został zmodyfikowany podczas transportu. Jednakże, podobnie jak w przypadku kodów uwierzytelniających wiadomości, nie mówi nic o tym, kiedy m zostało wysłane, a ataki polegające na powtórzeniu wiadomości są nadal możliwe (patrz sekcja 4.2).

Założenie, że strony są w stanie uzyskać legalną kopię klucza publicznego S , oznacza, że S jest w stanie przesyłać co najmniej jedną wiadomość (tj. samo pk) w sposób niezawodny i uwierzytelny. Jeśli jednak jesteśmy w stanie niezawodnie przesyłać wiadomości, to po co nam w ogóle schemat podpisu?

Odpowiedź jest taka, że niezawodna dystrybucja pk jest zadaniem trudnym i kosztownym, ale użycie schematu podpisu oznacza, że wystarczy to wykonać

raz, po czym można później niezawodnie wysłać nieograniczoną liczbę wiadomości. Co więcej, jak omówimy w podrozdziale 12.7, same schematy podpisów służą do zapewnienia niezawodnej dystrybucji innych kluczów publicznych.

Służą zatem jako centralne narzędzie do tworzenia „infrastruktury klucza publicznego” w celu rozwiązywania problemu dystrybucji kluczów.

Bezpieczeństwo schematów podpisów. W przypadku stałego klucza publicznego pk wygenerowanego przez podpisującego S , fałszerstwem jest wiadomość m wraz z ważnym podpisem σ , gdzie m nie zostało wcześniej podpisane przez S . Bezpieczeństwo schematu podpisu oznacza, że przeciwnik nie powinien być w stanie wyprowadzić fałszerstwa nawet jeśli uzyska podpisy pod wieloma innymi wybranymi przez siebie wiadomościami. Jest to bezpośrednia analogia z definicją bezpieczeństwa kodów uwierzytelniających wiadomości i odsyłamy czytelnika do Sekcji 4.2 w celu uzyskania motywacji i dalszej dyskusji.

Przedstawiamy teraz formalną definicję bezpieczeństwa, która jest zasadniczo taka sama jak Definicja 4.2. Niech $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ będzie schematem sygnatur i rozważmy następujący eksperyment dla przeciwnika A i parametru n :

Charakterystyczny eksperyment $\text{Sig-forge}_{A,\Pi}(n)$:

1. Uruchomiono $\text{Gen}(1^n)$ w celu uzyskania kluczów (pk, sk).
2. Przeciwnik A otrzymuje pk i dostęp do wyroczni $\text{Sign}_k(\cdot)$.
Następnie przeciwnik wyprowadza (m, σ) . Niech Q oznacza zbiór wszystkich zapytań, które A zadał swojej wyroczni.
3. A powiedzie się wtedy i tylko wtedy, gdy (1) $\text{Vrfy}_{pk}(m, \sigma) = 1$ i (2) $m \in Q$.
W tym przypadku wynik eksperymentu definiuje się jako 1.

DEFINICJA 12.2 Schemat podpisu $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ jest egzystencjalnie niemożliwy do podrobienia w wyniku adaptacyjnego ataku wybraną wiadomością lub po prostu bezpieczny, jeśli dla wszystkich probabilistycznych przeciwników A w czasie wielomianowym istnieje pomijalna funkcja zaniedbania taka, że :

$$\Pr[\text{Sig-forge}_{A,\Pi}(n) = 1] \leq \text{negl}(n).$$

12.3 Paradygmat hash-and-sign

Podobnie jak w przypadku szyfrowania kluczem publicznym i kluczem prywatnym, „rodzime” schematy podpisów są o rząd wielkości mniej wydajne niż kody uwierzytelniające wiadomości. Na szczęście, podobnie jak w przypadku szyfrowania hybrydowego (patrz rozdział 11.3), możliwe jest uzyskanie funkcjonalności podpisów cyfrowych asymptotycznym kosztem operacji na kluczu prywatnym, przynajmniej dla wystarczająco długich wiadomości. Można to zrobić za pomocą metody mieszania i znaku, omówionej dalej.

Intuicja stojąca za podejściem opartym na haszu i znaku jest prosta. Założymy, że mamy schemat podpisu dla wiadomości o długości i chcemy podpisać (dłuższą) wiadomość $m \in \{0, 1\}^n$. Zamiast podpisywać samo m , możemy zamiast tego użyć funkcji skrótu H , aby zaszyfrować wiadomość do skrótu o stałej długości o długości n , a następnie podpisać wynikowy skrót. Podejście to jest dokładnie analogiczne do podejścia opartego na hash i MAC omówionego w sekcji 5.3.1.

BUDOWA 12.3

Niech $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ będzie schematem podpisu dla komunikatów o długości (n) i niech $\Pi_H = (\text{Gen}_H, H)$ będzie funkcją skrótu o długości wyjściowej (n) . Skonstruuj schemat podpisu $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ w następujący sposób:

- Gen : na wejściu 1^n , uruchom $\text{Gen}(1^n)$, aby otrzymać (pk, sk) i uruchom $\text{Gen}_H(1^n)$, aby otrzymać s ; klucz publiczny to pk , s , a klucz prywatny to sk , s .
- Znak : na wejściu klucz prywatny sk , si komunikat $m \in \{0, 1\}^n$, wyjście $\sigma = \text{Sign}_{sk}(H s(m))$.
- Vrfy : na wejściu klucz publiczny pk , s , wiadomość $m \in \{0, 1\}^n$, i a sygnatura σ , wyjście 1 wtedy i tylko wtedy, gdy $\text{Vrfy}_{pk}(H s(m), \sigma) = 1$.

Paradygmat hash-and-sign.

TWIERDZENIE 12.4 Jeśli Π jest schematem bezpiecznego podpisu dla wiadomości o długości, a Π_H jest odporny na kolizje, to Konstrukcja 12.3 jest schematem bezpiecznego podpisu (dla wiadomości o dowolnej długości).

Dowód tego twierdzenia jest prawie identyczny z dowodem Twierdzenia 5.6.

12.4 Podpisy RSA

Rozważanie konkretnych schematów podpisów rozpoczynamy od omówienia schematów opartych na założeniu RSA.

12.4.1 Zwykły RSA

Najpierw opisujemy prosty schemat podpisu oparty na RSA. Chociaż schemat jest niepewny, służy jako przydatny punkt wyjścia.

Jak zwykle, niech GenRSA będzie algorytmem ppt, który na wejściu 1^n wyprowadza a

moduł N będący iloczynem dwóch n-bitowych liczb pierwszych (z wyjątkiem znikomego prawdopodobieństwa) wraz z liczbami całkowitymi e, d spełniającymi $ed = 1 \pmod{\varphi(N)}$. Generowanie klucza w zwykłym RSA polega po prostu na uruchomieniu GenRSA i wygenerowaniu N, e jako klucza publicznego i N, d jako klucza prywatnego. Aby N, podpisać wiadomość m Z podpisując oblicza $\sigma := [md \pmod{N}]$. Weryfikacja podpisu σ na wiadomości m w odniesieniu do klucza publicznego N, e odbywa się poprzez sprawdzenie, czy $m^e \equiv \sigma \pmod{N}$. Patrz Konstrukcja 12.5. σ

BUDOWA 12.5

Niech GenRSA będzie jak w tekście. Zdefiniuj schemat podpisu w następujący sposób:

- Gen: na wejściu 1n uruchom GenRSA(1n), abytrzymać (N, e, d). Klucz publiczny to N, e, a klucz prywatny to N, d.
- Znak: na wejściu klucz prywatny sk = N, d i komunikat m Z N, obliczyć podpis

$$\sigma := [m^d \pmod{N}].$$

- Vrfy: na wejściu klucz publiczny pk = N, e, wiadomość m Z N, I podpis σ Z N, wyjście 1 wtedy i tylko wtedy, gdy

$$m^e \equiv \sigma \pmod{N}.$$

Zwykły schemat podpisu RSA.

Łatwo zauważyc, że weryfikacja legalnie wygenerowanego podpisu jest od tego czasu zawsze odnosci sukcesy

$$e\sigma = (md) \equiv m[ed \pmod{\varphi(N)}] = m1 = m \pmod{N}.$$

Można by oczekiwac, że ten schemat będzie bezpieczny, ponieważ dla przeciwnika znającego jedynie klucz publiczny N, e obliczenie prawidłowego podpisu w wiadomości m wydaje się wymagać rozwiązania problemu RSA (ponieważ podpis jest dokładnie pierwiastkiem eth z m). Niestety, to rozumowanie jest błędne. Po pierwsze, założenie RSA implikuje jedynie trudność obliczenia podpisu (to znaczy obliczenia pierwiastka eth) jednolitej wiadomości m; nie mówi nic o trudnościach w obliczeniu podpisu na niejednorodnym m lub na jakiejś wiadomości m wybranej przez atakującego. Co więcej, założenie RSA nie mówi nic o tym, co osoba atakująca może zrobić, gdy pozna podpisy w innych wiadomościach. Poniższe przykłady pokazują, że obie te obserwacje prowadzą do ataków na zwykły schemat podpisu RSA.

Atak bez wiadomości. Pierwszy opisany przez nas atak generuje fałszerstwo przy użyciu samego klucza publicznego, bez uzyskania jakichkolwiek podpisów od uprawnionego podpisującego. Atak działa w następujący sposób: mając klucz publiczny pk = N, e, wybierz jednolite σ Z i oblicz m := [σe mod N]. Następnie wprowadź fałszerstwo N

(m, o). Od razu widać, że σ jest ważnym podpisem na m i jest to fałszerstwo, ponieważ właściciel klucza publicznego nie wydał żadnych podpisów. Dochodzimy do wniosku, że zwykły schemat podpisu RSA nie spełnia definicji 12.2.

Można argumentować, że nie stanowi to „realistycznego” ataku, ponieważ przeciwnik „nie ma kontroli” nad komunikatem, dla którego fałszuje ważny podpis. Nie ma to znaczenia, jeśli chodzi o Definicję 12.2 i już omawialiśmy (w Rozdziale 4), dlaczego niebezpieczne jest przyjmowanie jakiegokolwiek semantyki w przypadku wiadomości, które będą uwierzytelniane przy użyciu dowolnego schematu kryptograficznego. Co więcej, przeciwnik ma pewną kontrolę nad m: na przykład wybierając wiele jednakowych wartości σ może (z dużym prawdopodobieństwem) uzyskać m z kilkoma bitami ustawnionymi w pożądany sposób. Wybierając ośmiorówkę sposobów, może być również możliwe wpłynięcie na wynikowy komunikat, dla którego zostanie wygenerowany fałszerstwo.

Podrabianie podpisu pod dowolną wiadomością. Bardziej szkodliwy atak na zwykły schemat podpisu RSA wymaga, aby przeciwnik uzyskał dwa podpisy od osoby podpisującej, ale pozwala mu na wystawienie sfałszowanego podpisu na dowolnej wybranej przez siebie wiadomości. Założmy, że przeciwnik chce sfałszować podpis w wiadomości m ∈ Z w odniesieniu do klucza publicznego pk = N, tzn. Przeciwnik wybiera dowolne m₁, m₂ ∈ Z różne od m takie, że m = m₁ · m₂ mod N.

Otrzymuje wówczas sygnatury σ₁, σ₂ odpowiednio na m₁, m₂. Na koniec wyprowadza σ := [σ₁ · σ₂ mod N] jako ważny podpis na m. To działa, ponieważ

$$e^{\sigma} (\sigma_1 \cdot \sigma_2) \equiv e^{(m_1 \cdot m_2) \text{mod } N} \equiv e^m = 2$$

wykorzystując fakt, że σ₁, σ₂ są ważnymi podpisami na m₁, m₂.

Możliwość sfałszowania podpisu na dowolnej wiadomości jest druzgocąca. Niemniej jednak można argumentować, że taki atak jest nierealistyczny, ponieważ przeciwnik nie będzie w stanie przekonać podpisującego do podpisania dokładnych wiadomości m₁ i m₂. Powtórzę raz jeszcze, że nie ma to znaczenia w kontekście definicji 12.2. Co więcej, niebezpieczne jest przyjmowanie założeń na temat tego, jakie wiadomości osoba podpisująca może chcieć podpisać, a jakich nie. Na przykład klient może użyć schematu podpisu do uwierzytelnienia na serwerze poprzez podpisanie losowego wezwanego wysłanego przez serwer. W tym przypadku złośliwy serwer byłby w stanie uzyskać podpis pod dowolną wybraną przez siebie wiadomością. Mówiąc bardziej ogólnie, przeciwnik może wybrać m₁ i m₂ jako „uzasadnione” wiadomości, które podpisujący zgodzi się podpisać. Na koniec należy zauważać, że atak można uogólnić: jeśli przeciwnik uzyska ważne podpisy na q dowolnych wiadomościach M = {m₁, ..., m_q}, wówczas przeciwnik może wyprowadzić ważny podpis na dowolnej z 2q - q innych wiadomości uzyskanych poprzez wzięcie iloczynów podzbiorów M (o wielkości róźnej od 1).

12.4.2 RSA-FDH i PKCS #1 v2.1

Można spróbować zapobiec atakom z poprzedniej sekcji, dokonując transformacji wiadomości przed ich podpisaniem. Oznacza to, że podpisujący określi teraz jako część swojego klucza publicznego (deterministyczna) funkcję H

z pewnymi właściwościami kryptograficznymi (opisanymi poniżej) mapowanie wiadomości do Z^N ; podpis w wiadomości m będzie wynosił $\sigma := [H(m)]^d \mod N$, a weryfikacja podpisu σ na wiadomości m będzie odbywać się poprzez sprawdzenie, czy $e \cdot \sigma \equiv H(m) \mod N$. Patrz Konstrukcja 12.6.

BUDOWA 12.6

Niech GenRSA będzie jak w poprzednich sekcjach i skonstruuj schemat podpisu w następujący sposób:

- Gen: na wejściu $1n$, uruchom GenRSA($1n$), aby obliczyć (N, e, d) . The klucz publiczny to N, e , a klucz prywatny to N, d .
W ramach generowania klucza funkcja $H : \{0, 1\}^n \rightarrow Z$ ale Z jest specyficzne, pozostawiamy to ukryte.
- Znak: na wejściu klucz prywatny N, d i wiadomość $m \in \{0, 1\}^n$, obliczać

$$\sigma := [H(m)]^d \mod N.$$
- Vrfy: po wprowadzeniu klucza publicznego N, e , wiadomości m i podpisu σ , wyjście 1 wtedy i tylko wtedy, gdy $e \cdot \sigma \equiv H(m) \mod N$.

Schemat podpisu RSA-FDH.

Jakich właściwości potrzebuje H , aby ta konstrukcja była bezpieczna?

Aby zapobiec atakowi bez wiadomości, atakujący nie powinien mieć możliwości rozpoczęcia od σ , obliczenia $\hat{m} := [\sigma e \mod N]$, a następnie znalezienia wiadomości m takiej, że $H(m) = \hat{m}$. Oznacza to w szczególności, że H powinno być w pewnym sensie trudne do odwrócenia. Aby zapobiec drugiemu atakowi, potrzebujemy H , które nie dopuszcza „relacji multiplikatywnych”, czyli dla którego trudno znaleźć trzy komunikaty m, m_1, m_2 z $H(m) = H(m_1) \cdot H(m_2) \mod N$. Wreszcie znalezienie kolizji w H musi być trudne: jeśli $H(m_1) = H(m_2)$, to m_1 i m_2 mają tę samą sygnaturę i fałszerstwo staje się banalne.

Nie jest znany sposób wyboru H , aby można było wykazać bezpieczeństwo konstrukcji 12.6. Można jednak udowodnić bezpieczeństwo, jeśli H jest modelowany jako losowa wyczynka, która równomiernie odwzorowuje swoje dane wejściowe. Zschemat w tym przypadku nazywany jest schematem podpisu RSA pełnej domeny skrótu (RSA-FDH). Można sprawdzić, czy funkcja losowa tego rodzaju spełnia wymagania omówione w poprzednim akapicie: funkcja losowa (o dużym zakresie) jest trudna do odwrócenia, nie ma łatwych do znalezienia relacji multiplikatywnych i jest odporna na kolizje. Oczywiście to nieformalne rozumowanie nie wyklucza wszystkich możliwych ataków, ale poniższy dowód bezpieczeństwa tak.

Zanim przejdziemy do dowodu formalnego, podamy trochę intuicji. Naszym celem jest udowodnienie, że jeśli problem RSA jest trudny w porównaniu z GenRSA, to RSA-FDH jest bezpieczny, gdy H jest modelowany jako losowa wyczynka. Rozważamy najpierw zabezpieczenie przed atakiem bez wiadomości, tj. gdy przeciwnik A nie może tego zażądać

podpisy. Tutaj przeciwnik ogranicza się do zadawania zapytań do losowej wyroczni i zakładamy bez utraty ogólności, że A zawsze zadaje dokładnie q (różnych) zapytań do H i że jeśli przeciwnik wyprowadza fałszerstwo (m, σ), to wcześniej pytał m do H .

Załóżmy, że istnieje skuteczny przeciwnik A , który przeprowadza atak bez wiadomości i wysyła dokładnie q zapytań do H . Konstruujemy wydajny algorytm A rozwiązujący problem RSA w odniesieniu do GenRSA. Biorąc pod uwagę dane wejściowe (N, e, y) , algorytm A uruchamia A na kluczu publicznym $pk = N, e$. Niech m_1, \dots, m_q oznaczają q (odrębne) zapytania, które A kieruje do H . Nasz algorytm A odpowiada na losowe zapytania Oracle A z jednolitymi elementami Z z wyjątkiem jednego zapytania —powiedzmy i -tego zapytania, wybranego jednakowo spośród zapytań Oracle A —na to odpowiada samo y . Zauważ, że z punktu widzenia A na wszystkie jego zapytania losowej wyroczni odpowiadają jednolite elementy Z (przypomnijmy, że y jest również jednolite, chociaż nie jest wybierane przez A), więc A nie ma informacji o i . Czy więcej, widok A , gdy jest uruchamiany jako podprogram przez A , jest identyczny z widokiem A podczas ataku na oryginalny schemat podpisu.

Jeśli A wyprowadza fałszerstwo (m, σ), to, ponieważ $m \in \{m_1, \dots, m_q\}$, z prawdopodobieństwem umiejętności $1/q$ będziemy mieli $m = m_i$. W tym wypadku,

$$e^\sigma = H(m) = H(m_i) = y \bmod N$$

i A może wyprowadzić σ jako rozwiązanie danej instancji RSA (N, e, y) . Dochodzimy do wniosku, że jeśli A wyprowadza fałszerstwo z prawdopodobieństwem ϵ , to A rozwiązuje problem RSA z prawdopodobieństwem ϵ/q . Ponieważ q jest wielomianem, dochodzimy do wniosku, że ϵ musi być pomijalne, jeśli problem RSA jest trudny w porównaniu z GenRSA.

Trudniejsze jest postępowanie w przypadku, gdy przeciwnik może żądać podpisów pod wybranymi przez siebie wiadomościami. Komplikacja pojawia się, ponieważ nasz algorytm A nie zna wykładownika deszyfrowania d , a mimo to musi teraz obliczyć prawidłowe podpisy w wiadomościach odpytywanych przez A do swojej wyroczni podpisującej. Wydaje się to niemożliwe (aby móc nawet sprzeczeć!), dopóki nie uświadomimy sobie, że A może poprawnie obliczyć podpis w wiadomości m , o ile ustawa $H(m)$ na wartość $[\sigma e \bmod N]$ dla znanej wartości σ . (Tutaj używamy faktu, że losowa wyrocznia jest „programowalna”). Jeśli σ jest jednolite, to $[\sigma e \bmod N]$ jest również jednolite, a więc losowa wyrocznia jest nadal „poprawnie” emulowana przez A .

Powyższa intuicja zostaje sformalizowana w dowodzie:

TWIERDZENIE 12.7 Jeśli problem RSA jest trudny w porównaniu z GenRSA, a H jest modelowany jako losowa wyrocznia, wówczas Konstrukcja 12.6 jest bezpieczna.

DOWÓD Niech $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ oznacza Konstrukcję 12.6 i niech A będzie probabilistycznym przeciwnikiem w czasie wielomianowym. Zakładamy bez utraty ogólności, że jeśli A żąda podpisu w wiadomości m lub wysyła fałszywe zapytanie (m, σ) , to wcześniej odpytywał m do H . Niech $q(n)$ będzie wielomianem górnej granicy liczby zapytań A powodując H w przypadku parametru bezpieczeństwa n ; zakładamy bez utraty ogólności, że A zadaje dokładnie $q(n)$ różne zapytania do H .

Dla wygody podajemy etapy eksperymentu $\text{Sig-forgeA}, \Pi(n)$:

1. $\text{GenRSA}(1n)$ jest uruchamiane w celu otrzymania (N, e, d) . Wybrano funkcję losową $H : \{0, 1\}^m \rightarrow \mathbb{Z}$.
2. Przeciwnik A ma dane $pk = N, e$ i może zapytać H oraz wyrocznię podpisującą $\text{Sign}_{N,d}(\cdot)$, która po wprowadzeniu wiadomości $m \in \mathbb{Z}^D$ zwraca $\sigma := [H(m) \bmod N]$
3. A wychodzi (m, σ) , gdzie wcześniej nie prosił o podpis na m. Wynik eksperymentu wynosi 1 wtedy i tylko wtedy, gdy $\sigma^{mi} = H(m) \bmod N$.

Definiujemy zmodyfikowany eksperiment $\text{Sig-forge A}, \Pi(n)$, w którym na początku zgaduje się, która wiadomość (spośród q wiadomości, które A odpytuje H) będzie odpowiadać ostatecznemu fałszemu (jeśli w ogóle) wynikowi przez A:

1. Wybierz jednolity j $\in \{1, \dots, Q\}$.
2. $\text{GenRSA}(1n)$ jest uruchamiane w celu otrzymania (N, e, d) . Wybrano funkcję losową $H : \{0, 1\}^m \rightarrow \mathbb{Z}$.
3. Przeciwnik A ma dane $pk = N, e$ i może zapytać H oraz wyrocznię podpisującą $\text{Sign}_{N,d}(\cdot)$, która po wprowadzeniu wiadomości $m \in \mathbb{Z}^D$ zwraca $\sigma := [H(m) \bmod N]$
4. A wyprowadza (m, σ) , gdzie wcześniej nie zażądał sig. Wyjście $= H(m)^2 \bmod N$ i $j = i$. Eksperiment wynosi 1 wtedy i tylko wtedy, gdy σ

Ponieważ j jest jednolite i niezależne od wszystkiego innego, prawdopodobieństwo, że $j = i$ (nawet uzależnione od przypadku, gdy A generuje fałszerstwo) wynosi dokładnie $1/q$. Zatem $\Pr[\text{Sig-forge A}, \Pi(n) = 1] = \frac{1}{q(n)} \cdot \Pr[\text{Sig-forgeA}, \Pi(n) = 1]$.

Rozważmy teraz zmodyfikowany eksperiment $\text{Sig-forge A}, \Pi(n)$, w którym eksperiment zostaje przerwany, jeśli A kiedykolwiek zażąda podpisu w wiadomości m_j (gdzie m_j oznacza j-tą wiadomość odpytywaną do H, a j jest wybraną jednolitą wartością Na wstępie). Nie zmienia to prawdopodobieństwa, że wynik eksperymentu wyniesie 1, ponieważ jeśli A kiedykolwiek zażąda podpisu na m_j , wówczas nie będzie możliwe wygenerowanie fałszerstwa na m_j . W słowach,

$$\begin{aligned} \Pr[\text{Sig-forge A}, \Pi(n) = 1] &= \Pr[\text{Sig-forge A}, \Pi(n) = 1] \\ &= \underline{\Pr[\text{Sig-forgeA}, \Pi(n) = 1]} q(n). \end{aligned} \quad (12.1)$$

² Tutaj mi oznacza i-te zapytanie skierowane do H. Przypomnijmy, z założenia, że jeśli A żąda podpisu w wiadomości m , to musiał wcześniej zapytać m do H.

Na koniec rozważ następujący algorytm A rozwiązujący problem RSA:

Algorytm A: Jako dane wejściowe podaje się algorytm (N, e, y) .

1. Wybierz jednolity $j \in \{1, \dots, Q\}$.
2. Uruchom A na wejściu klucza publicznego $pk = N$, np. Przechowuj trójkę (\cdot, \cdot, \cdot) w tabeli, początkowo pustej. Wpis (mi, σ, yi) wskazuje, że A ustawił $H(mi) = yi$ i $\sigma = yi \text{ mod } N$.
 σ
3. Kiedy A zadaje i-te zapytanie losowej wyroczni $H(mi)$, odpowiedz na nie w następujący sposób:
 - Jeśli $i = j$, zwróć y jako odpowiedź na zapytanie.
 - W innym wypadku wybierz uniform $\sigma' \in \mathbb{Z}_{N^{\frac{m}{2}}} \text{ mod } N$, zwróć yi jako odpowiedź na zapytanie i zapisz (mi, σ', yi) w tabeli. σ'
4. Na koniec wykonywania A wyprowadza (m, σ) . Jeżeli $m = mj = y \text{ mod } N$, i $\sigma = \sigma'$ to wyprowadź σ .

Jest oczywiste, że A przebiega w probabilistycznym czasie wielomianowym. Założmy, że dane wejściowe (N, e, y) do A są generowane przez uruchomienie $\text{GenRSA}(1n)$ w celu uzyskania (N, e, d) , a jednolity j następuje wybranie N . Kluczową obserwacją jest to, że widok A, gdy jest uruchamiany jako podprogram $y = Z$ A jest identyczny z widokiem A w eksperymencie $\text{Sig-forge}_A, \Pi(n)$. W szczególności odpowiedzi na wszystkie zapytania Sign-Oracle są poprawne, a na każde zapytanie Random-Oracle_A , uruchamiane jako podprogram przez A, odpowiada jednolity element Z :

- Na zapytanie $H(mj)$ odpowiada się y , jednolitym elementem $Z \in \mathbb{Z}_{N^{\frac{m}{2}}} \text{ mod } N$.
- Na zapytania $H(mi)$ z $i = j$ odpowiada się $yi = [\sigma \text{ jest jednorodne w } \mathbb{Z}_{N^{\frac{m}{2}}} \text{ mod } N]$, gdzie $\sigma \in \mathbb{Z}_{N^{\frac{m}{2}}} \text{ mod } N$. Ponieważ potęgowanie do potęgi e jest równa jeden do jednego funkcji Z , yi również ma rozkład jednostajny.

Na koniec zauważ, że ilekroć eksperyment $\text{Sig-forge}_A, \Pi(n)$ dałby wynik 1, wówczas A wyprowadza poprawne rozwiązanie dla danej instancji RSA. Wynika to z faktu, że $\text{Sig-forge}_A, \Pi(n) = 1$ implikuje, że $j = i$ oraz $\sigma = H(mi) \text{ mod } N$. Teraz, gdy $j = i$, algorytm A nie przerywa $=$ dodatkowo $H(mi) = y$. Zatem $\sigma =$

³ Tutaj mi oznacza i-te zapytanie skierowane do H. Przypomnijmy, z założenia, że jeśli A żąda podpisu w wiadomości m, to musiał wcześniej zapytać m do H.

$H(m) = y \bmod N$, zatem σ jest pożądaną odwrotnością. Korzystając z równania (12.1), oznacza to, że

$$\begin{aligned} \Pr[\text{RSA-invA ,GenRSA}(n) = 1] &= \Pr[\text{Sig-forge A}, \Pi(n) = 1] \\ &= \frac{\Pr[\text{Sig-forgeA}, \Pi(n) = 1]}{q(n)}. \end{aligned} \quad (12.2)$$

Jeśli problem RSA jest trudny w porównaniu z GenRSA, istnieje pomijalna funkcja negl taka, że $\Pr[\text{RSA-invA ,GenRSA}(n) = 1] = \text{negl}(n)$. Ponieważ $q(n)$ jest wielomianem, z równania (12.2) wnioskujemy, że $\Pr[\text{Sig-forgeA}, \Pi(n) = 1]$ jest również nieistotne. To kończy dowód. ■

RSA PKCS #1 wersja 2.1. Standard RSA PKCS #1 v2.1 zawiera schemat podpisu, który można postrzegać jako wariant RSA-FDH. Nieco dokładniej standard definiuje schemat, w którym podpis na wiadomości jest zależny od soli (czyli losowej wartości) wybranej przez podpisującego w momencie generowania podpisu. Jeśli podpisujący ustali tę sól na NULL – jest to coś, na co pozwala standard – powstał schemat jest bardzo podobny do RSA-FDH.

Ważne jest, aby zakres H był (bliski) calementu Z_N ; w szczególności nie wystarczy po prostu pozwolić, aby H był „gotową” kryptograficzną funkcją skrótu, taką jak SHA-1. (Długość wyjściowa SHA-1 jest znacznie mniejsza niż długość modułów RSA stosowanych w praktyce.) Rzeczywiście, praktyczne ataki na konstrukcję 12.6 są znane, jeśli długość wyjściowa H jest zbyt mała (np. jeśli długość wyjściowa wynosi 160 bitów co miałoby miejsce, gdyby SHA-1 użyto bezpośrednio jako H). W schemacie podpisu PKCS #1 v2.1 H jest konstruowany poprzez wielokrotne zastosowanie podstawowej kryptograficznej funkcji skrótu.

12.5 Podpisy z problemu logarytmu dyskretnego

Schematy podpisów mogą być również oparte na założeniu logarytmu dyskretnego, chociaż założenie to nie nadaje się do podpisów tak łatwo, jak założenie RSA. W sekcji 12.5.1 opisujemy schemat podpisów wprowadzony przez Clausa Schnorra, którego bezpieczeństwo można udowodnić w modelu losowej wyroczni. W sekcji 12.5.2 opisujemy powszechnie stosowane schematy podpisów DSA i ECDSA.

12.5.1 Schemat podpisu Schnorra

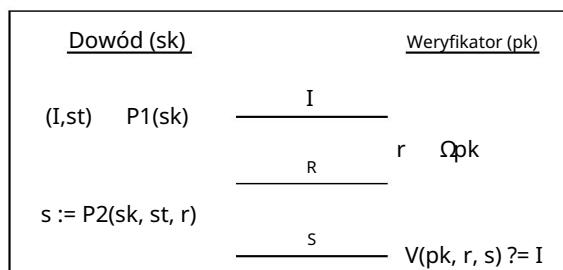
Intuicję leżącą u podstaw schematu podpisu Schnorra najlepiej wyjaśnić, odchodząc nieco od tematu, omawiając schematy identyfikacji (klucza publicznego). Następnie opisujemy transformację Fiata – Shamira, której można użyć do konwersji schematów identyfikacji na schematy podpisów w modelu losowej wyroczni. Wreszcie,

przedstawiamy schemat identyfikacji Schnorra - i odpowiadający mu schemat podpisu - oparty na problemie logarytmu dyskretnego.

Schematy identyfikacyjne

Schemat identyfikacji to interaktywny protokół, który pozwala jednej stronie udowodnić swoją tożsamość (tj. uwierzytelnić się) drugiej. Jest to zjawisko bardzo naturalne i obecnie powszechnie jest uwierzytelnianie się podczas logowania do serwisu internetowego. Stronę identyfikującą się (np. użytkownika) nazywamy „weryfikatorem”, a stronę weryfikującą tożsamość (np. serwer WWW) „weryfikatorem”. W tym przypadku interesuje nas ustalenie klucza publicznego, w którym weryfikator i weryfikator nie udostępniają z góry żadnych tajnych informacji (takich jak hasło); zamiast tego weryfikator zna tylko klucz publiczny weryfikatora. Pomyślne wykonanie protokołu identyfikacyjnego przekonuje weryfikatora, że komunikuje się z zamierzonym weryfikatorem, a nie z oszustem.

Rozważmy jedynie trójrundowe protokoły identyfikacyjne o określonej postaci, gdzie udowadniający jest określony przez dwa algorytmy P_1 , P_2 , a strona protokołu weryfikatora jest określona przez algorytm V . Prower uruchamia $P_1(sk)$ przy użyciu swojego klucza prywatnego sk w celu uzyskania komunikatu początkowego I wraz z pewnym stanem st i inicjuje protokół wysyłając I do weryfikatora. W odpowiedzi weryfikator wysyła wyzwanie r wybrane jednolicie z pewnego zbioru Ω_{pk} określonego przez klucz publiczny pk weryfikatora. Następnie dowódca uruchamia $P_2(sk, st, r)$, aby obliczyć odpowiedź, którą odsyła do weryfikatora. Na koniec weryfikator oblicza $V(pk, r, s)$ i akceptuje wtedy i tylko wtedy, gdy skutkuje to początkowym komunikatem I ; patrz rysunek 12.1. Oczywiście dla poprawności wymagamy, aby jeśli legalny weryfikator wykona protokół poprawnie, to weryfikator zawsze zaakceptował.



RYSUNEK 12.1: Trzyrundowy schemat identyfikacji.

Ze względów technicznych zakładamy, że schematy identyfikacji są „niezdegenerowane”, co intuicyjnie oznacza, że możliwych komunikatów początkowych I jest wiele i żaden nie ma dużego prawdopodobieństwa wysłania. Formalnie schemat nie jest zdegenerowany, jeśli dla każdego klucza prywatnego sk i dowolnej ustalonej wiadomości początkowej I , the

prawdopodobieństwo, że $P1(sk)$ wyprowadza I jest znikome. (Każdy schemat identyfikacji można w prosty sposób zmodyfikować, aby nie był zdegenerowany, wysyłając jednolity n -bitowy ciąg znaków wraz z komunikatem początkowym.)

Podstawowym wymogiem bezpieczeństwa schematu identyfikacji jest to, że przeciwnik, który nie zna tajnego klucza osoby sprawdzającej, nie powinien być w stanie oszukać weryfikatora, aby się zgodził. Powinno to obowiązywać nawet wtedy, gdy atakujący jest w stanie biernie podsłuchiwać wielokrotne (uczciwe) wykonania protokołu pomiędzy osobą sprawdzającą a weryfikatorem. Formalizujemy takie podsłuchiwanie za pośrednictwem wyroczni Transsk, która po wywołaniu bez żadnego wkładu przeprowadza uczciwe wykonanie protokołu i zwraca przeciwnikowi cały zapis (I, r, s) interakcji.

Niech $\Pi = (\text{Gen}, P1, P2, V)$ będzie schematem identyfikacji i rozważmy następujący eksperyment dla przeciwnika A i parametru n :

Eksperyment identyfikacyjny $\text{IdentA}, \Pi(n)$:

1. Uruchomiono $\text{Gen}(1n)$ w celu uzyskania kluczy (pk, sk) .
2. Przeciwnik A otrzymuje pk i dostęp do wyroczni Transsk
może wysyłać zapytania tak często, jak chce.
3. W dowolnym momencie eksperymentu A wysyła komunikat I .
Wybrano jednolite wyzwanie $r \in \Omega_{pk}$ i dano A , który odpowiedział kilkoma s . (A może w dalszym ciągu zwracać się do Transska nawet po otrzymaniu r .)
4. Doświadczenie daje 1 wtedy i tylko wtedy, gdy $V(pk, r, s) = I$.

DEFINICJA 12.8 Schemat identyfikacji $\Pi = (\text{Gen}, P1, P2, V)$ jest zabezpieczony przed atakiem pasywnym lub po prostu zabezpieczony, jeśli dla wszystkich probabilistycznych przeciwników A w czasie wielomianowym istnieje pomijalna funkcja negl taka, że:

$$\Pr[\text{IdentA}, \Pi(n) = 1] = \text{negl}(n).$$

Można także rozważyć mocniejsze koncepcje bezpieczeństwa, np. gdy przeciwnik może również przeprowadzać aktywne ataki na protokół, podszywając się pod weryfikator i ewentualnie wysyłając złośliwie wybrane wartości r . Nie będziemy tego potrzebować w naszym zastosowaniu do schematów podpisów.

Od schematów identyfikacyjnych do podpisów

Transformacja Fiata - Shamira (konstrukcja 12.9) umożliwia przekształcenie dowolnego (interaktywnego) schematu identyfikacji w (nieinteraktywny) schemat podpisu. Podstawową ideą jest, aby podpisujący pełnił rolę dowodzącego, samodzielnie uruchamiając protokół identyfikacyjny. Oznacza to, że aby podpisać wiadomość m , osoba podpisująca najpierw oblicza I , a następnie generuje wyzwanie r , stosując jakąś funkcję H do I i m . Następnie wyprowadza poprawną odpowiedź s . Sygnatura na m to (r, s) , co można zweryfikować poprzez (1) ponowne obliczenie $I := V(pk, r, s)$, a następnie (2) sprawdzenie, że $H(I, m) = r$.

BUDOWA 12.9

Niech $(\text{Genid}, \text{P1}, \text{P2}, \text{V})$ będzie schematem identyfikacyjnym i skonstruuje schemat podpisu w następujący sposób:

- Gen: na wejściu $1n$ po prostu uruchom $\text{Genid}(1n)$, aby uzyskać klucze pk , sk .
Klucz publiczny pk określa zestaw wyzwań Ω_{pk} . W ramach generowania klucza podawana jest funkcja $H : \{0, 1\}^* \rightarrow \Omega_{\text{pk}}$, ale pozostawiamy to implicit.
- Znak: na wejściu klucz prywatny sk i komunikat $m \in \{0, 1\}^*$,
Do:
 1. Oblicz $(I, st) = \text{P1}(\text{sk})$.
 2. Oblicz $r := H(I, m)$.
 3. Oblicz $s := \text{P2}(\text{sk}, st, r)$.

Wyprowadź podpis (r, s) .

- Vrfy: po wprowadzeniu klucza publicznego pk , wiadomości m i podpisu (r, s) , oblicz $I := \text{V}(\text{pk}, r, s)$ i wyprowadź 1 wtedy i tylko wtedy, gdy $H(I, m) = r$.

Transformacja Fiata – Shamira.

Podpis (r, s) jest „powiązany” z konkretną wiadomością m , ponieważ r jest funkcją zarówno I , jak i m ; zmiana m powoduje zatem zupełnie inne r . Jeśli H jest modelowane jako losowe dane wejściowe mapujące wyrocznię równomiernie na Ω_{pk} , wówczas wyzwanie r jest jednolite; intuicyjnie, przeciwnikowi (nieznającemu sk) będzie tak samo trudno znaleźć prawidłowy podpis (r, s) w wiadomości m , jak podszywać się pod dowódcę w uczciwym wykonaniu protokołu.

Intuicja ta zostaje sformalizowana w dowodzie następującego twierdzenia.

TWIERDZENIE 12.10 Niech Π będzie schematem identyfikacyjnym i niech Π' będzie schematem sygnatur powstającym po zastosowaniu do niego transformaty Fiata – Shamira. Jeśli Π jest bezpieczne, a H jest modelowane jako losowa wyrocznia, to Π' jest bezpieczne.

DOWÓD Niech A będzie probabilistycznym przeciwnikiem działającym w czasie zapytań o schemat podpisu Π' , wielomianowym, atakującym z $q = q(n)$ górną granicą liczby Π , które A wykonuje do H . Przyjmujemy szereg upraszczających założeń bez utraty ogólności. Po pierwsze, zakładamy, że A kieruje dowolne zapytanie do H tylko raz. Zakładamy również, że po otrzymaniu podpisu (r, s) na wiadomości m z $\text{V}(\text{pk}, r, s) = I$, przeciwnik A nigdy nie zadaje pytania $H(I, m)$ (ponieważ wie, że odpowiedzią będzie r). Na koniec zakładamy, że jeśli A wysyła sfałszowany podpis (r, s) w wiadomości m z $\text{V}(\text{pk}, r, s) = I$, to A wcześniej pytał $H(I, m)$.

Konstruujemy wydajnego przeciwnika A' , który używa A jako podprogramu i atakuje schemat identyfikacji Π :

Algorytm A:

Algorytmowi podano pk i dostęp do wyroczni $Transsk$.

1. Wybierz jednolity $j \in \{1, \dots, Q\}$.

2. Uruchom A (pk). Odpowiedz na jego pytania w następujący

sposób: Kiedy A wykona i-te zapytanie losowej wyroczni $H(I_i, m)$, odpowiedz na nie w następujący sposób:

- Jeśli $i = j$, wyprowadź I_j i otrzymaj w zamian wyzwanie r .
Zwróć r do A jako odpowiedź na jego zapytanie. •

Jeśli $i = j$, wybierz jednorodne $r \in pk$ i zwróć r jako odpowiedź na zapytanie.

Kiedy A prosi o podpis na m , odpowiedz w następujący sposób: (a)

Zapytaj o $Transsk$, aby uzyskać transkrypcję (I, r, s) uczciwego wykonania protokołu. (b) Zwróć podpis (r, s) .

s).

3. Jeśli A wysyła sfałszowany podpis (r, s) w wiadomości m , oblicz $I := V(pk, r, s)$ i sprawdź, czy $(I, m) = (I_j, m_j)$. Jeśli więc wyprowadź s . W przeciwnym razie przerwij.

Widok A uruchamianego jako podprogram przez A w eksperymencie $IdentA, \Pi(n)$ jest prawie identyczny z widokiem A w eksperymencie $Sig-forgeA, \Pi(n)$. Rzeczywiście, na wszystkie zapytania H wysypane przez A odpowiada się z jednolitą wartością z Ωpk , a na wszystkie zapytania o podpisywane wysypane przez A odpowiada się ważnymi podpisami o prawidłowym rozkładzie. Jedyna różnica między poglądami polega na tym, że gdy A jest uruchamiane jako podprogram przez A, możliwe jest, że wystąpią niespójności w odpowiedziach, które A otrzyma na swoje zapytania do H: w szczególności dzieje się tak, jeśli A kiedykolwiek odpowie na zapytanie o podisanie wiadomość m przy użyciu transkryptu (I, r, s) , dla którego $H(I, m)$ jest już zdefiniowane (to znaczy, że A wcześniej pytał (I, m) do H) i $H(I, m) = r$. Jeśli jednak Π nie jest zdegenerowane, dzieje się to tylko z nikomu prawdopodobieństwem. Zatem prawdopodobieństwo, że A wygeneruje fałszerstwo, gdy zostanie uruchomione jako podprogram przez A, wynosi $Sig-forgeA, \Pi(n) - negl(n)$ dla jakiejś pomijalnej funkcji $negl$.

Rozważmy wykonanie eksperymentu $IdentA, \Pi(n)$, w którym A wysyła sfałszowany podpis (r, s) na wiadomość m i niech $I := V(pk, r, s)$. Ponieważ j jest jednolite i niezależne od wszystkiego innego, prawdopodobieństwo, że $(I, m) = (I_j, m_j)$ (nawet uzależnione od przypadku, gdy A generuje fałszerstwo) wynosi dokładnie $1/q$. (Przypomnijmy, zakładamy, że jeśli A wyśle sfałszowany podpis (r, s) w wiadomości m z $V(pk, r, s) = I$, to A wcześniej zapytał $H(I, m)$.) Gdy oba zdarzenia się zdarzyć, A skutecznie podszywa się pod dowódcę. Rzeczywiście, A wysyła I_j jako swój początkowy komunikat, otrzymuje w odpowiedzi wezwanie r i odpowiada s .

Ale $H(I_j, m_j) = r$ oraz (ponieważ sfałszowany podpis jest ważny) $V(pk, r, s) = I$. Składając wszystko razem, widzimy to

$$\Pr[IdentA, \Pi(n) = 1] = \frac{1}{q} \cdot \Pr[Sig-forgeA, q(n), \Pi(n) = 1] - negl(n)$$

Lub

$$\Pr[\text{Sig-forgeA}_{\Pi}(n) = 1] \leq q(n) \cdot \Pr[\text{IdentA}, \Pi(n) = 1] + \text{negl}(n).$$

Jeżeli Π jest pewne, to $\Pr[\text{IdentA}, \Pi(n) = 1]$ jest nieistotne; ponieważ $q(n)$ jest wielomianem, oznacza to, że $\Pr[\text{Sig-forgeA}(n) = 1]$ jest również nieistotne. Ponieważ A było arbitralne, oznacza to, że Π jest bezpieczne. ■

Schemat identyfikacji Schnorra

Schemat identyfikacji Schnorra opiera się na twardości problemu logarytmu dyskretnego. Niech G będzie algorytmem czasu wielomianowego, który przyjmuje na wejściu $1n$ i (z wyjątkiem prawdopodobnie znikomego prawdopodobieństwa) wyprowadza opis grupy cyklicznej G, jej rząd q (przy $q = n$) i generator g. Aby wygenerować klucze, dowód uruchamia $G(1n)$ w celu otrzymania (G, q, g) , wybiera jednorodny $x \in Z_q$ i ustawia $y := g^x$; klucz publiczny to G, q, g, y , a klucz prywatny x . Aby wykonać protokół (patrz rysunek 12.2), dowód rozpoczyna się od wybrania jednolitego $k \in \mathbb{Z}_{q-1}$ jako wiadomość początkową. Weryfikator wybiera i wysyła jednolite wyzwanie $r \in Z_q$; w odpowiedzi dowód $r =? I$.

oblicza $s := [rx + k \bmod q]$. Weryfikator akceptuje wtedy i tylko wtedy, gdy $g^{s \cdot j} = g^r \cdot g^k = g^r \cdot y = g^r \cdot g^x = g^r \cdot g^y = g^r \cdot I$. Poprawność zachodzi, ponieważ

$$G^s \cdot j = g^{rx+k} \cdot (g^x)^r \cdot g^k = g^r \cdot g^x = g^r \cdot y = g^r \cdot I$$

Zauważ, że I jest jednolite w G, więc schemat nie jest zdegenerowany.

Przed przedstawieniem dowodu zapewniamy intuicję wysokiego poziomu. Pierwszą ważną obserwacją jest to, że bierne podsłuchiwanie nie jest pomocne dla atakującego. Dzieje się tak dlatego, że atakujący może samodzielnie symulować transkrypcje uczciwych egzekucji, opierając się wyłącznie na kluczu publicznym i bez znajomości klucza prywatnego. Aby to zrobić, atakujący po prostu odwraca kolejność kroków: najpierw wybiera jednolite i niezależne $r, s \in Z_q$, a następnie ustawia $I := g^r \in W'$ uczciwym transkryptem (I, r, s). Wiadomość początkowa I jest jednolitym elementem G, wyzwanie jest niezależnym, jednolitym elementem Z_q , a s jest wówczas jednoznacznie określone jako $s = \log_g(I \cdot y^r)$. Symulowane transkrypcje utworzone przez atakującego

Dowód(x)	Weryfikator(G, q, g, y)
$k \in Z_q$	
$ja := g^k$	I
	$r \in Z_q$
	R
$s := [rx + k \bmod q]$	
	s
	sprawdź, czy $g^{s \cdot j} = g^r \cdot I$

RYSUNEK 12.2: Wykonanie schematu identyfikacji Schnorra.

mają ten sam rozkład: $r \equiv Zq$ jest jednorodny, a ponieważ s jest jednorodne w Zq i niezależne od r , widzimy, że I jest jednorodne w G i niezależne od r .

Wreszcie s jest jednoznacznie określone jako spełniające to samo ograniczenie co poprzednio.

Dzięki temu można skutecznie założyć, że atakując schemat identyfikacji, atakujący w ogóle nie podsłuchuje uczciwych egzekucji.

Zatem ograniczyliśmy się do atakującego, który otrzymuje klucz publiczny y , wysyła początkową wiadomość I , w odpowiedzi otrzymuje jednolite wyzwanie r , a następnie musi wysłać odpowiedź s , dla której $g^s = I$. Nieformalnie, jeśli atakujący jest w stanie aby to zrobić z dużym prawdopodobieństwem, musi w szczególności być w stanie obliczyć prawidłowe odpowiedzi s_1, s_2 na co najmniej dwa różne wyzwania $r_1, r_2 \in Zq$. Notatka

$$g^{s_1} \cdot y^{r_1} = ja = rz^{s_2} \cdot y^{r_2},$$

i tak G . Oznacza to jednak, że atakujący (który, jak pamiętamy, jest w stanie wygenerować s_1 w odpowiedzi na r_1 i s_2 w odpowiedzi na r_2) może pośrednio obliczyć logarytm dyskretny

$$\log y = [(s_1 - s_2) \cdot (r_1 - r_2)]^{-1} \pmod{q},$$

sprzeczne z założoną trudnością problemu logarytmu dyskretnego.

TWIERDZENIE 12.11 Jeżeli problem logarytmu dyskretnego jest trudny w stosunku do G , to schemat identyfikacji Schnorra jest bezpieczny.

DOWÓD Niech Π oznacza schemat identyfikacji Schnorra i niech A będzie zdecydowanym przeciwnikiem atakującym ten schemat. Konstruujemy następujący algorytm ppt A rozwiązujący problem logarytmu dyskretnego względem G :

Algorytm A: Algorytm
ma dane wejściowe G, q, g, y .

1. Uruchom $A(pk)$, odpowiadając na wszystkie jego zapytania do Transska, jak opisano w podanej wcześniej intuicji.
2. Kiedy A wyprowadza I , wybierz jednolite $r_1 \in Zq$ jako wyzwanie.
Daj r_1 A , który odpowie s_1 .
3. Uruchom $A(pk)$ drugi raz (od początku), stosując tę samą losowość co poprzednio, z wyjątkiem jednolitego i niezależnego $r_2 \in Zq$. Ostatecznie A odpowiada s_2 .
4. Jeśli $g^{s_1} \cdot y^{-r_1} = I$ i $g^{s_2} \cdot y^{-r_2} \not\equiv I$ i $r_1 = r_2$, następnie wyprowadź $\log y \pmod{q}$. W $[(s_1 - s_2) \cdot (r_1 - r_2)]^{-1}$ przeciwnym razie nie wyświetlaj niczego.

Rozważenie pojedynczego przebiegu A jako podprogramu A ness, niech woznacza losowość użytego w tym wykonaniu, z wyjątkiem samego wyzwania. Zatem wobejmuję dowolną losowość używaną przez G , wybór (nieznanego) klucza prywatnego x , dowolną losowość używaną przez samego A oraz losowość używaną przez A podczas odpowiadania na zapytania

do Transska. Zdefiniuj $V(\omega r)$ jako równe 1 wtedy i tylko wtedy, gdy A poprawnie odpowie na wyzwanie r , gdy w pozostałej części wykonania wykorzystana zostanie losowość ω . Dla dowolnego $\omega \stackrel{\text{def}}{=} \Pr[V(\omega r) = 1]$; przy ustalonym ω jest to prawdopodobieństwo ustalonego wzdefiniu δ zamiast wyboru wyzwania r , na które A odpowie poprawnie.

Zdefiniowaliśmy, $\delta(n) = \Pr[\text{IdentA}, \Pi(n) = 1]$. Od czasu symulacji Transska że $\delta(n)$ wyrocznia jest doskonała

$$\delta(n) = \Pr_{\omega} [V(\omega r) = 1] = \Pr_{\omega} [\omega \cdot \delta\omega]$$

Co więcej, intuicja poprzedzająca dowód pokazuje, że A poprawnie oblicza logarytm dyskretny y , ilekroć A powiedzie się dwa razy i $r_1 = r_2$. Zatem:

$$\begin{aligned} \Pr[DLogA, G(n) = 1] &= \Pr_{\omega, r_1, r_2} [V(\omega r_1) = V(\omega r_2) \quad r_1 = r_2] \\ &= \Pr_{\omega} [\Pr_{\omega} [V(\omega r_1) = V(\omega r_2)] \cdot \Pr_{\omega} [r_1 = r_2]] \\ &= \Pr_{\omega} [\omega \cdot \delta\omega]^2 \cdot 1/kw \\ &= \delta(n)^2 \cdot 1/q, \end{aligned}$$

wykorzystując nierówność Jensena⁴ w przedostatnim kroku. Jeśli problem logarytmu dyskretnego jest trudny w stosunku do G , wówczas $\Pr[DLogA, G(n) = 1]$ jest pomijalny. Ponieważ $1/q$ jest nieistotne (ponieważ $q = n$), oznacza to, że $\delta(n)$ jest również nieistotne, a więc Π jest bezpiecznym schematem identyfikacji. ■

Schemat podpisu Schnorra uzyskuje się poprzez zastosowanie Fiata – Shamira przekształcić do schematu identyfikacji Schnorra. Zobacz Konstrukcja 12.12.

BUDOWA 12.12

Niech G będzie takie, jak opisano w tekście.

- Gen: uruchom $G(1n)$, aby otrzymać (G, q, g) . Wybierz uniform $x \in \mathbb{Z}_q$ i klucz publiczny $y := g^x$ prywatny to x , a klucz publiczny to (G, q, g, y) . określono \mathbb{Z}_q , Jako część generowania klucza, funkcja $H : \{0, 1\}^*$, ale pozostawiamy to ukryte.
- Znak: po wprowadzeniu klucza prywatnego x i wiadomości $m \in \{0, 1\}^*$, wybierać uniform $k \in \mathbb{Z}_q$ i zbiór $I := g^k$, po którym $r := H(I, m)$, następuje $s := [rx + k \bmod q]$. Wyprawdź podpis (r, s) .
- Vrfy: po wprowadzeniu klucza publicznego (G, q, g, y) , komunikatu m i podpisu (r, s) , oblicz $I := g^{s \cdot j + r}$ i wyjście 1, jeśli $H(I, m) \stackrel{?}{=} R$.

Schemat podpisu Schnorra.

4Mówi o tym nierówność Jensena $\Pr_{\omega} [\omega \cdot b] \stackrel{2}{>} \Pr_{\omega} [\omega \cdot b] \stackrel{1}{>} \Pr_{\omega} [\omega \cdot b]$ dla pozytywnego $\{\omega\}$.

12.5.2 DSA i ECDSA

Algorytm podpisu cyfrowego (DSA) i algorytm podpisu cyfrowego na krzywej eliptycznej (ECDSA) opierają się na problemie logarytmu dyskretnego w różnych klasach grup. Istnieją w jakiejś formie od 1991 roku i oba są uwzględnione w bieżącym standardzie podpisu cyfrowego (DSS) wydanym przez NIST.

Obydwa schematy opierają się na wspólnym szablonie i można je postrzegać jako zbudowane na podstawie podstawowego schematu identyfikacji (patrz poprzednia sekcja). Niech G będzie grupą cykliczną rzędu pierwszego q z generatorem g . Rozważmy następujący schemat identyfikacji, w którym klucz prywatny sprawdzającego to x , a klucz publiczny to (G, q, g, y) , gdzie $y = g^x$

$x:$

1. Dowód wybiera jednorodność $k \in \mathbb{Z}_q$ i wysyła $I := g^k$.

2. Weryfikator wybiera i wysyła jako wyzwanie jednolite $a, r \in \mathbb{Z}_q$.

3. Prowadzący wysyła $s := [k]^{-1} \cdot (a + xr) \bmod q$ jako odpowiedź.

as 1 4. Weryfikator akceptuje, jeśli $s = 0$ oraz $g \cdot y^{rs-1} \equiv I$.

Uwaga $s = 0$, chyba że $a = xr \bmod q$, co zachodzi z znakomym prawdopodobieństwem. Zakładając $s = 0$, odwrotność $s^{-1} \bmod q$ istnieje i

$$as^{-1}rs^{-1} \equiv 1 \quad as^{-1}xr^{-1} \equiv 1 \quad (a+xr)s^{-1} \equiv g \cdot y \equiv g \cdot g \equiv g \quad 1 \quad (a+xr) \cdot k \cdot (a+xr)^{-1} \equiv g^{-1} \equiv ja$$

Widzimy zatem, że poprawność zachodzi z prawie znakomym prawdopodobieństwem.

Można wykazać, że ten schemat identyfikacji jest bezpieczny, jeśli problem logarytmu dyskretnego jest trudny w stosunku do G . Po prostu szkicujemy argument, zakładając znajomość wyników z poprzedniej sekcji. Przede wszystkim można symulować transkrypcje uczciwych egzekucji: w tym celu wystarczy wybrać $as^{-1}rs^{-1} \equiv 1$. (To już nie jest jednolite $a, r \in \mathbb{Z}_q$, ale s^{-1} jest wystarczająco blisko.) Co nastepnie ustanie więcej, jeśli atakujący wyślę początkowy komunikat I , dla którego może dać poprawne odpowiedzi $s_1, s_2 \in \mathbb{Z}_q$ na różne wyzwania $(a, r_1), (a, r_2)$ wówczas

$$G^{as^{-1}} \cdot r^{r_1s^{-1}} \equiv ja = dz \quad as^{-2} \cdot r^{r_2s^{-2}} \equiv 1,$$

$a(s, r, r_1, r_2)$ i $g^{2 \log_2 h} \bmod q$ można obliczyć jak w poprzedniej sekcji. To samo dotyczy sytuacji, gdy atakujący udzieli poprawnych odpowiedzi na różne wyzwania $(a_1, r), (a_2, r)$.

Schematy podpisu DSA/ECDSA są konstruowane poprzez „zwinięcie” powyższego schematu identyfikacji do nieinteraktywnego algorytmu uruchamianego przez osobę podpisującą. Jednak w przeciwnieństwie do transformacji Fiata – Shamira, transformacja tutaj jest przeprowadzana w następujący sposób (patrz Konstrukcja 12.13):

- Ustaw $\alpha := H(m)$, gdzie m jest podpisaną wiadomością, a H jest kryptograficzna funkcja skrótu.

- Ustaw $r := F(I)$ dla (określonej) funkcji $F : G \rightarrow Z_q$. W tym przypadku F jest „prostą” funkcją, która nie ma działać jak przypadkowa wyrocznia.

Funkcja F zależy od grupy G_m , która z kolei zależy od schematu. W DSA G przyjmuje się jako podgrupę rzędu q dla p pierwszego

(por. sekcja 8.3.3) i $F(I) \stackrel{\text{def}}{=} [I \bmod q]$. W ECDSA G jest podgrupą rzędu q grupy o krzywej eliptycznej $E(Z_p)$, dla p prim.5 Przypomnijmy sobie z sekcji 8.3.4, że każdy element takiej grupy można przedstawić jako parę $(x, y) \in Z_p \times Z_p$.

Funkcja F w tym przypadku jest zdefiniowana jako $F(x, y) \stackrel{\text{def}}{=} [x \bmod q]$.

BUDOWA 12.13

Niech G będzie jak w tekście.

- Gen: na wejściu $1n$, uruchom $G(1n)$, aby otrzymać (G, q, g) . Wybierz $x \in Z_q$ i ustaw $y := g^x \bmod q$. Klucz publiczny to G, q, g, y i klucz prywatny to x .
W ramach generowania klucza określane są dwie funkcje $H : \{0, 1\}^* \rightarrow Z_q$ i $F : G \rightarrow Z_q$, ale pozostawiamy to implicit.
- Znak: po wprowadzeniu klucza prywatnego x i wiadomości $m \in \{0, 1\}^*$, wybierz uniform $k \in \{0, 1\}^*$ i ustaw $r := F(g^k \bmod q)$. Następnie oblicz $s := (H(m) + xr) \bmod q$. (Jeśli $r = 0$ lub $s = 0$, zacznij od nowa [kaśnie wybór k].) Wypisz podpis (r, s) .
- Vrfy: na wejściu klucz publiczny G, q, g, y , wiadomość $m \in \{0, 1\}^*$, sygnatura (r, s) z $r, s \in Z_q$, wyjście 1 wtedy i tylko wtedy, gdy

$$R_g y \quad H(m)s - 1 \bmod q \stackrel{?}{=} F(r)$$

DSA i ECDSA – abstrakcja.

Zakładając trudność problemu logarytmu dyskretnego, można udowodnić, że DSA i ECDSA są bezpieczne, jeśli H i F są modelowane jako losowe wyrocznie. Jednakże, jak omówiliśmy powyżej, chociaż model losowej wyroczni może być uzasadniony dla H , nie jest to odpowiedni model dla F . Nie są znane żadne dowody bezpieczeństwa dla konkretnych wyborów F w standardzie. Niemniej jednak DSA i ECDSA były używane i badane przez dziesięciolecia i nie stwierdzono żadnych ataków.

Właściwe pokolenie k. Schematy DSA/ECDSA określają, że podpisujący powinien podczas obliczania podpisu wybrać jednolite $k \in Z_q$. Nieprawidłowy wybór k (np. z powodu złego generowania liczb losowych) może prowadzić do katastrofalnych skutków. Na początek, jeśli atakujący może przewidzieć wartość k użytą do obliczenia podpisu (r, s) w wiadomości m , to może obliczyć

⁵ECDSA dopuszcza również krzywe eliptyczne nad innymi polami, ale przypadek pól pierwszych omówiliśmy jedynie w podrozdziale 8.3.4.

klucz prywatny osoby podpisującej. Jest to prawdą, ponieważ $s = k^{-1} \cdot (H(m) + xr) \bmod q$, k i jeśli k jest znane, jedyną niewiadomą jest klucz prywatny x.

Nawet jeśli k jest nieprzewidywalne, osoba atakująca może obliczyć klucz prywatny osoby podpisującej, jeśli to samo k zostanie kiedykolwiek użyte do wygenerowania dwóch różnych podpisów. Atakujący może łatwo stwierdzić, kiedy to nastąpi, ponieważ wtedy r również się powtarza. Powiedzmy, że $(r, s1)$ i $(r, s2)$ to sygnatury odpowiednio wiadomości $m1$ i $m2$. Następnie

$$\begin{aligned}s1 &= k^{-1} \cdot (H(m1) + xr) \bmod q \\s2 &= k^{-1} \cdot (H(m2) + xr) \bmod q.\end{aligned}$$

Odejmowanie daje $s1 - s2 = k^{-1} \cdot (H(m1) - H(m2)) \bmod q$, z którego można obliczyć k; mając k, atakujący może określić klucz prywatny x, jak w poprzednim akapicie. Właśnie ten atak został wykorzystany przez hakerów do wydobycia głównego klucza prywatnego z konsoli Sony PlayStation (PS3) w 2010 roku.

12.6 *Podpisy z funkcji skrótu

Co ciekawe – i być może nieco zaskakujące – schematy podpisów można konstruować w oparciu o kryptograficzne funkcje skrótu, bez polegania na założeniach teorii liczb. (Jest to przeciwnieństwo szyfrowania kluczem publicznym, które wydaje się wymagać poważnych problemów z jakąś strukturą algebraiczną.) W tej sekcji przyjrzymy się takim konstrukcjom. Schematy, które zobaczymy, są również interesujące pod tym względem, że nie opierają się na przypadkowych wyroczniach, w przeciwieństwie do wszystkich konstrukcji, które opisaliśmy wcześniej w tym rozdziale.

12.6.1 Schemat podpisu Lamporta

Nasze badanie schematów podpisów opartych na funkcjach skrótu rozpoczynamy od rozważenia stosunkowo słabego pojęcia schematów podpisów jednorazowych. Nieformalnie takie schematy są „bezpieczne”, o ile dany klucz prywatny służy do podpisania tylko jednej wiadomości. Schematy spełniające tę koncepcję bezpieczeństwa mogą być odpowiednie dla niektórych zastosowań, a także służyć jako przydatne „cegielki” do osiągnięcia silniejszych koncepcji bezpieczeństwa, jak zobaczymy w następnej sekcji.

Niech $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ będzie schematem podpisu i rozważ następujące kwestie eksperymentu dla przeciwnika A i parametru n:

Jednorazowy eksperyment z podpisem Sig-forge1-time A, $\Pi(n)$:

1. Uruchomiono $\text{Gen}(1n)$ w celu uzyskania kluczy (pk, sk).
2. Przeciwnik A otrzymuje pk i zadaje pojedyncze zapytanie m swojej wyroczni $\text{Sign}_{\text{sk}}(\cdot)$. Następnie A wyprowadza (m, o) za pomocą $m = m$.
3. Wynik eksperymentu definiuje się jako 1 wtedy i tylko wtedy, gdy $\text{Vrfy}_{\text{pk}}(m, o) = 1$.

DEFINICJA 12.14 Schemat podpisu $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ jest egzystencjalnie niemożliwy do podrobienia w przypadku ataku pojedynczą wiadomością lub jest schematem jednorazowego bezpiecznego podpisu, jeśli dla wszystkich probabilistycznych przeciwników A w czasie wielomianowym istnieje znikomy funkcja negl taka, że:

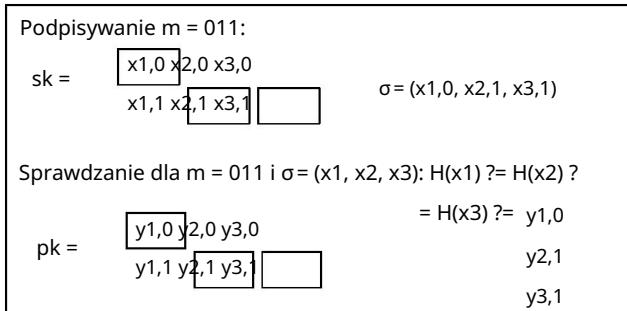
$$\Pr[\text{Sig-forge}_1^{\text{t-ADP}}(\Pi) = 1] \leq \text{negl}(n).$$

Leslie Lamport pokazał konstrukcję schematu jednorazowego bezpiecznego podpisu w 1979 roku. Ilustrujemy pomysł dla przypadku podpisywania wiadomości 3-bitowych. Niech H będzie kryptograficzną funkcją skrótu. Klucz prywatny składa się z sześciu jednakowych wartości $x_1, 0, x_1, 1, x_2, 0, x_2, 1, x_3, 0, x_3, 1 \in \{0, 1\}^n$, a odpowiadający mu klucz publiczny zawiera wyniki uzyskane poprzez zastosowanie H do każdego z tych elementów. Klucze te można wizualizować jako tablice dwuwymiarowe:

$$\begin{array}{ll} \text{pk} = & \begin{array}{l} y_1, 0 \ y_2, 0 \ y_3, 0 \\ y_1, 1 \ y_2, 1 \ y_3, 1 \end{array} & \text{sk} = & \begin{array}{l} x_1, 0 \ x_2, 0 \ x_3, 0 \\ x_1, 1 \ x_2, 1 \ x_3, 1 \end{array} \end{array}.$$

Aby podpisać wiadomość $m = m_1m_2m_3$ (gdzie $m_i \in \{0, 1\}$), osoba podpisująca uwalnia odpowiedni obraz wstępny x_i, m_i dla każdego bitu wiadomości; podpis σ składa się z trzech wartości $(x_1, m_1, x_2, m_2, x_3, m_3)$. Weryfikacja odbywa się w sposób naturalny: przedstawia się podpisem kandydata (x_1, x_2, x_3) na wiadomości $m = m_1m_2m_3$, akceptujemy wtedy i tylko wtedy, gdy $H(x_i) = y_i, m_i$ dla $1 \leq i \leq 3$.

Pokazano to graficznie na rysunku 12.3, a ogólny przypadek – dla wiadomości dowolnej długości – opisano formalnie w Konstrukcji 12.15.



RYSUNEK 12.3: Schemat Lamporta użyty do podpisania wiadomości $m = 011$.

Po zaobserwowaniu podpisu na wiadomości osoba atakująca, która chce sfałszować podpis na jakiekolwiek innej wiadomości, musi znaleźć obraz wstępny jednego z trzech „nieużywanych” elementów klucza publicznego. Jeżeli H jest jednokierunkowe (patrz Definicja 8.72), to znalezienie takiego obrazu wstępnego jest trudne obliczeniowo.

TWIERDZENIE 12.16 Niech będzie dowolnym wielomianem. Jeśli H jest funkcją jednokierunkową, wówczas Construction 12.15 jest schematem jednorazowego bezpiecznego podpisu.

BUDOWA 12.15

Niech $H : \{0, 1\}^n \rightarrow \{0, 1\}^n$ będzie funkcją. Skonstruuj schemat podpisu dla wiadomości o długości $= (n)$ w następujący sposób:

- Gen: na wejściu $1n$ postępuj w następujący sposób dla $i = \{1, \dots, n\}$:

1. Wybierz uniform $x_{i,0}, x_{i,1} \in \{0, 1\}^n$ 2. Oblicz $y_{i,0}, y_{i,1} := H(x_{i,0}, x_{i,1})$.

Klucz publiczny pk i klucz prywatny sk to

$$pk = \begin{matrix} y_{1,0} \\ y_{2,0} \\ \vdots \\ y_{n,0} \end{matrix} \quad sk = \begin{matrix} x_{1,0} \\ x_{2,0} \\ \vdots \\ x_{n,0} \end{matrix} .$$

- Znak: po wprowadzeniu klucza prywatnego sk jak wyżej i komunikatu $m \in \{0, 1\}^n$ gdzie $m = m_1 \cdots m_n$, wypisz podpis $(x_1, m_1, \dots, x_n, m_n)$.
- Vrfy: po wprowadzeniu klucza publicznego pk jak wyżej, komunikat $m \in \{0, 1\}^n$ z $m = m_1 \cdots m_n$ i podpisem $\sigma = (x_1, \dots, x_n)$, wyjście 1 wtedy i tylko jeśli $H(x_i) = y_{i,m_i}$ dla wszystkich $i = 1, \dots, n$.

Schemat podpisu Lamporta.

DOWÓD Niech $= (n)$ w całości. Jak zauważono przed chwilą, kluczowa obserwacja jest następująca: powiedzmy, że atakujący A żąda podpisu w wiadomości m i rozważ każdą inną, wiadomość $m' = m$. Musi istnieć co najmniej jedna pozycja $\{1, \dots, n\}$ na którym m i m' na ~~największą możliwą odległość~~ ~~właściwie największą~~ różnicę w i -tym położeniu klucza publicznego. Ponieważ H jest jednokierunkowe, powinno to być niewykonalne. Teraz $y_{i,b}$ formalizujemy tę intuicję.

Niech Π oznacza schemat Lamporta i niech A będzie probabilistycznym przeciwnikiem wielomianowym w czasie. W konkretnym wykonaniu Sig-forge1-tym (n) niech m oznaczy wiadomość, której podpisu żąda A (zakładamy bez utraty ogólności, że A zawsze żąda podpisu na wiadomości) i niech (m, o) będzie końcowym wyjściem A. Mówimy, że A wyprowadza fałszerstwo w (i, b) jeśli $Vrfypk(m, o) = 1$ (tzn. komunikaty m i m' różnią się na j -tym położeniu - a ponadto $m_i = m'_i$ i $m_j = b \neq m'_j$). Należy zauważać, że ilekroć A swoim i -tym położeniu - a ponadto $m_i = m'_i$ i $m_j = b \neq m'_j$ A wyprowadza fałszerstwo, w pewnym momencie (i, b) wyprowadza fałszerstwo.

Rozważ następujący algorytm ppt, który próbuje odwrócić H :

Algorytm I:

Algorytm otrzymuje na wejściu $1n$ i $y = \{1, \dots, n\}$

1. Wybierz mundur, tj. $i \in \{1, \dots, n\}$. Ustaw $y_{i,b} := y$.
2. Dla wszystkich $i = \{1, \dots, n\}$ i $b \in \{0, 1\}$ gdzie $(i, b) = (i, b')$:
 - Wybierz uniform $x_{i,b} \in \{0, 1\}^n$ i ustaw $y_{i,b} := H(x_{i,b})$.
3. Uruchom A na wejściu $pk := \begin{matrix} y_{1,0} \\ y_{2,0} \\ \vdots \\ y_{n,0} \end{matrix}$.

4. Kiedy A prosi o podpis w wiadomości m:

- Jeśli $m_i = b$, następnie przerywam wykonanie.
- W przeciwnym razie zwracam podpis $\sigma = (x_1, m_1, \dots, x_i, m_i)$.

5. Gdy A wyprowadza (m, ϕ) przy $\sigma = (x_1, \dots, x_i)$:

- Jeśli A wygeneruje fałszerstwo w (i, b) , a następnie wypisz x_i .

Ilekroć A wyprowadza fałszerstwo w (i, dane, b) , algorytm I udaje się odwrócić wejściowe y. Interesuje nas prawdopodobieństwo, że tak się stanie, gdy dane wejściowe I zostaną wygenerowane poprzez wybranie uniforma $x \in \{0, 1\}^n$ i ustawienie $y := H(x)$ (por. Definicja 8.72). Wyobraźmy sobie „eksperyment umysłowy”, w którym na początku dostaję x, ustawiam $x_i, b := x$, a potem zawsze zwracam sygnaturę A w kroku 4 (tzn. nawet jeśli $m = b$). Widok A uruchamianego jako podprogram przez I w tym eksperymencie mentalnym ma rozkład identyczny jak widok A w eksperymencie Sig-forge1-time $b \in \{0, 1\}^n$. Został wybrany równomiernie na początku eksperimentu, a pogląd A jest niezależny od tego. Ponieważ prawdopodobieństwo, że A wyprowadzi fałszerstwo w (i, b) , uwarunkowane faktem, że A w ogóle wyprowadzi fałszerstwo, wynosi co najmniej $1/2$. (Dzieje się tak, ponieważ fałszerstwo podpisu implikuje fałszerstwo co najmniej w jednym punkcie (i, b) .) Ponieważ jest $2(n)$ punktów, prawdopodobieństwo fałszerstwa w (i, b) wynosi co najmniej $1/2^n$. Dochodzimy do wniosku, że w tym eksperymencie myślowym prawdopodobieństwo, że A wyprowadza

- $\Pr[\text{Sig-forge1-time fałszerstwo przy } (i, A, \Pi(n)) = 1]$.

(i, b) jest co najmniej $\frac{1}{2^n}$

najmniej. Wracając do prawdziwego eksperimentu z udziałem I, jak opisano na początku, kluczową obserwacją jest to, że prawdopodobieństwo, że A wyprowadza fałszerstwo w (i, b) pozostaje taka sama, niezmienione. Dzieje się tak dlatego, że eksperiment mentalny i eksperiment rzeczywisty pokrywają się, jeśli A wyprowadza fałszerstwo w (i, b) . Oznacza to, że eksperymenty tylko się różnią, ale jeśli tak się dzieje, to (z definicji) nie jest możliwe, aby A doszło do sytuacji, gdy $m = b$. Zatem prawdopodobieństwo, że A wyprowadza $\Pr[\text{Sig-forge1-time } A, \Pi(n) = 1]$. Innymi słowy, fałszerstwo w (i, b) jest nadal co najmniej

$$\Pr[\text{InwertI}, H(n) = 1] \geq \frac{1}{2^n} \cdot \Pr[\text{Sig-forge1-time } A, \Pi(n) = 1].$$

Ponieważ H jest funkcją jednokierunkową, istnieje pomijalna funkcja negl taka, że

$$\text{negl}(n) = \Pr[\text{InvertI}, H(n) = 1].$$

Ponieważ jest to wielomian, oznacza to, że $\Pr[\text{Sig-forge1-time } A, \Pi(n)] \neq 1$, co kończy dowód. ■

DODATEK 12.17 Jeżeli istnieją funkcje jednokierunkowe, to dla każdego wielomianu istnieje schemat jednorazowego bezpiecznego podpisu dla wiadomości o długości .

12.6.2 Podpisy łańcuchowe

Istotną wadą jest oczywiście możliwość podpisania tylko jednej wiadomości danym kluczem prywatnym. Pokazujemy tutaj podejście oparte na odpornych na kolizje funkcjach skrótu, które pozwala podpisującemu podpisać dowolną liczbę wiadomości kosztem utrzymania stanu, który musi być aktualizowany po wygenerowaniu każdego podpisu. W sekcji 12.6.3 omawiamy bardziej efektywny wariant tego podejścia (który nadal wymaga stanu), a następnie opisujemy, w jaki sposób ta zmodyfikowana konstrukcja może stać się bezstanowa. Wynik pokazuje, że schematy sygnatur spełniające definicję 12.2 można skonstruować w oparciu o odporne na kolizje funkcje skrótu.

Najpierw definiujemy schematy podpisów, które pozwalają podpisującemu utrzymać stan który jest aktualizowany po złożeniu każdego podpisu.

DEFINICJA 12.18 Stanowy schemat podpisu to krotka probabilistycznych algorytmów czasu wielomianowego (Gen , Sign , Vrfy) spełniających poniższe wymagania:

1. Algorytm generowania klucza Gen przyjmuje na wejściu parametr bezpieczeństwa n i wyprowadza $(\text{pk}, \text{sk}, s_0)$. Nazywa się je odpowiednio kluczem publicznym, kluczem prywatnym, 1 i stanem początkowym. Zakładamy, że każdy pk i sk ma długość co najmniej n i że n można wyznaczyć z pk , sk .
2. Algorytm podpisywania Sign przyjmuje na wejściu klucz prywatny sk o wartości $si \in \{1, \dots, t\}$ i wiadomość $m \in \{0, 1\}^n$. Wyprowadza sygnaturę σ o wartość si .
3. Deterministyczny algorytm weryfikacji Vrfy przyjmuje jako dane wejściowe klucz publiczny pk , wiadomość m i podpis σ . Wychodzi trochę b .

Wymagamy, aby dla każdego n każdy $(\text{pk}, \text{sk}, s_0)$ wynik $\text{Gen}(1^n)$ i dowolny $mes \in \{0, 1\}^n$ = $1, \dots, t$, to dla i , jeśli iteracyjnie obliczymy $(\sigma_i, si) = \text{Sign}_{\text{sk}}, si = 1(m_i)$ miedzy m_1, \dots, m_t , dla i każdego $i \in \{1, \dots, t\}$, utrzymuje się, że $\text{Vrfy}_{\text{pk}}(m_i, \sigma_i) = 1$.

Podkreślamy, że weryfikator nie musi znać stanu podpisującego, aby zweryfikować podpis; w rzeczywistości w niektórych programach podpisujący musi zachować tajemnicę stanu, aby zachować bezpieczeństwo. Schematy podpisów, które nie zachowują stanu (jak w definicji 12.1), nazywane są bezstanowymi, aby odróżnić je od schematów stanowych. Oczywiście preferowane są schematy bezstanowe (choćże schematy stanowe mogą nadal być potencjalnie przydatne). Wprowadzamy podpisy stanowe jako odszkocznę do ewentualnej konstrukcji bezpieczeństwa.

Bezpieczeństwo schematów podpisów stanowych jest dokładnie analogiczne do definicji 12.2, z jedyną subtelnością polegającą na tym, że wyrocznia podpisująca zwraca tylko podpis (a nie stan) i że wyrocznia podpisująca aktualizuje stan przy każdym wywołaniu.

Dla dowolnego wielomianu $t = t(n)$ możemy łatwo skonstruować stanowy schemat podpisu „ t -time-safe”. (Definicja bezpieczeństwa byłaby tu oczywistym uogólnieniem definicji 12.14.) Możemy to zrobić po prostu pozwalając, aby klucz publiczny (odpowiednio klucz prywatny) składał się z niezależnie wygenerowanych kluczy publicznych (odpowiednio kluczy prywatnych) dla dowolnego schematu jednorazowego podpisu bezpiecznego; tj,

zestaw $\text{pk} := \text{pk}_1, \dots, \text{pk}_t$ i $\text{sk} := \text{sk}_1, \dots, \text{sk}_t$ gdzie każdy (pk_i, sk_i) jest niezależnie wygenerowaną parą kluczy dla jakiegoś schematu jednorazowego bezpiecznego podpisu.

Stanem jest licznik i początkowo ustawiony na 1. Aby podpisać wiadomość m przy użyciu klucza prywatnego sk_i i aktualnego stanu t , oblicz $\sigma = \text{Sign}_i(m)$ (czyli wygeneruj podpis na m przy użyciu klucza prywatnego sk_i) i wyjście (σ, t) ; stan jest aktualizowany do $i := i + 1$. Ponieważ stan zaczyna się od 1, oznacza to, że i -ta wiadomość jest podpisana za pomocą sk_i . Weryfikacja podpisu (σ, t) na wiadomości m odbywa się poprzez sprawdzenie, czy σ jest podpisem ważnym na m w odniesieniu do pk_i .

Ten schemat jest bezpieczny, jeśli jest używany do podpisywania t wiadomości, ponieważ każdy klucz prywatny podstawowego schematu jednorazowego zabezpieczenia jest używany do podpisywania tylko jednej wiadomości.

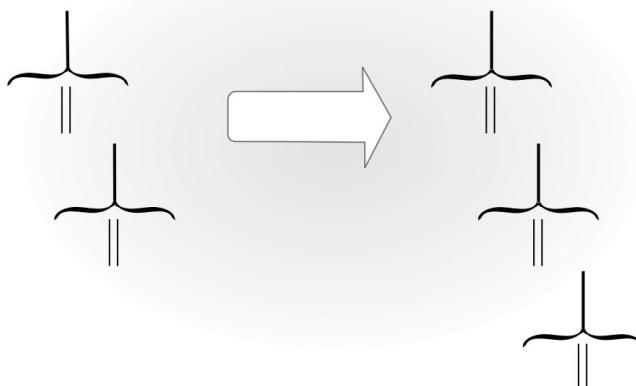
Jak opisano, podpisy mają stałą długość (tj. niezależną od t), ale klucz publiczny ma długość liniową w t . Można zamienić długość klucza publicznego i podpisu, prosząc podpisującego o obliczenie drzewa Merkle $h := \text{MT}_t(\text{pk}_1, \dots, \text{pk}_t)$ (patrz sekcja 5.6.2) na podstawie t podstawowych kluczy publicznych z schematem jednorazowego zabezpieczenia. Oznacza to, że kluczem publicznym będzie teraz t, h , a podpis na i -tej wiadomości będzie zawierał (σ, i) , tak jak poprzednio, wraz z i -tą wartością pk_i i dowodem t , że jest to poprawna wartość odpowiadająca h . (Weryfikacja odbywa się w sposób naturalny.) Klucz publiczny ma teraz stały rozmiar, a długość podpisu rośnie jedynie logarytmicznie wraz z t .

Ponieważ t może być dowolnym wielomianem, dlaczego poprzednie schematy nie dają nam rozwiązania, którego szukamy? Główną wadą jest to, że wymagają ustalenia górnej granicy t liczby wiadomości, które można podpisać, z góry, w momencie generowania klucza. Jest to potencjalnie poważne ograniczenie, ponieważ po osiągnięciu górnej granicy konieczne byłoby wygenerowanie i rozpowszechnienie nowego klucza publicznego. Zamiast tego wolelibśmy mieć jeden, stały klucz publiczny, którego można używać do podpisywania nieograniczonej liczby wiadomości.

Niech $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ będzie schematem jednorazowego bezpiecznego podpisu. W schemacie, który właśnie opisaliśmy (omijając optymalizację drzewa Merkle), osoba podpisująca wykonuje wywołania Gen w celu uzyskania kluczy publicznych $\text{pk}_1, \dots, \text{pk}_t$ i zawiera każdy z nich w swoim rzeczywistym kluczu publicznym pk_i . Osoba podpisująca może wówczas podpisywać co najwyżej t wiadomości. Możemy działać lepiej, stosując schemat „oparty na łańcuchu”, w którym osoba podpisująca generuje na bieżąco dodatkowe klucze publiczne, w razie potrzeby.

W schemacie łańcuchowym klucz publiczny składa się tylko z pojedynczego klucza publicznego pk_1 wygenerowanego przy użyciu Gen , a klucz prywatny to tylko powiązany klucz prywatny sk_1 . Aby podpisać pierwszą wiadomość m_1 , osoba podpisująca najpierw generuje nową parę kluczy $(\text{pk}_2, \text{sk}_2)$ przy użyciu Gen , a następnie podpisuje zarówno m_1 , jak i pk_2 przy użyciu sk_1 , aby otrzymać $\sigma_1 = \text{Sign}_{\text{sk}_1}(m_1, \text{pk}_2)$. Wyprowadzany podpis zawiera zarówno pk_2 , jak i σ_1 , a podpisujący dodaje $(m_1, \text{pk}_2, \text{sk}_2, \sigma_1)$ do swojego bieżącego stanu. Ogólnie rzecz biorąc, gdy nadejdzie czas podpisania i -tej wiadomości, osoba podpisująca będzie przechowywała $\{(m_j, \text{pk}_j+1, \text{sk}_j+1, \sigma_j)\}_{j=1}^{i-1}$ jako część swojego państwa. Aby podpisać i -tą wiadomość m_i , najpierw generuje nową parę kluczy $(\text{pk}_i+1, \text{sk}_i+1)$ za pomocą Gen , a następnie podpisuje mi i pk_i+1 za pomocą sk_i , aby uzyskać podpis $\sigma_i = \text{Sign}_{\text{sk}_i}(m_i, \text{pk}_i+1)$.

Rzeczywisty podpis, który jest wyprowadzany, zawiera pk_i+1 , a także wartości $\sigma_i \{m_j, \text{pk}_j+1, \sigma_j\}_{j=1}^{i-1}$. Podpisujący następnie dodaje $(m_i, \text{pk}_i+1, \text{sk}_i+1, \sigma_i)$ do swojego stanu. Graficzne przedstawienie tego procesu przedstawiono na rysunku 12.4.



RYSUNEK 12.4: Podpisy łańcuchowe: sytuacja przed i po podpisaniu trzeciej wiadomości m3.

Do weryfikacji podpisu $(pk_i+1, \sigma_i, \{mj, pk_j+1, \sigma_j\}_{j=1}^{i-1})$ w komunikacie $m = m_i$ w odniesieniu do klucza publicznego pk_1 odbiorca sprawdza każde połączenie pomiędzy kluczem publicznym pk_j a kolejnym kluczem publicznym pk_{j+1} w łańcuchu, a także połączenie pomiędzy ostatnim kluczem publicznym pk_i oraz m . Oznacza to, że weryfikacja daje 1 wtedy i tylko wtedy, gdy

$$\text{Vrfypk}_j(m_j|pk_j+1, \sigma_j) \geq 1 \text{ dla wszystkich } j \in \{1, \dots, I\}. \text{ (Patrz rysunek 12.4.)}$$

Nietrudno przekonać się – przynajmniej na poziomie intuicyjnym – że ten schemat podpisu jest egzystencjalnie niemożliwy do podrobienia w przypadku adaptacyjnego ataku wybranej wiadomości (niezależnie od tego, ile wiadomości jest podpisanych). Nieformalnie wynika to po raz kolejny z faktu, że każda para kluczy (pk_i, sk_i) jest używana do podpisania tylko jednej „wiadomości”, gdzie w tym przypadku „wiadomość” jest w rzeczywistości parą wiadomość/klucz publiczny $m|pk_i+1$. Ponieważ w następnej sekcji udowodnimy bezpieczeństwo bardziej wydajnego schematu, nie udowadniajmy tutaj bezpieczeństwa schematu opartego na łańcuchu.

W schemacie łańcuchowym każdy klucz publiczny pk_i służy do podpisywania zarówno wiadomości, jak i innego klucza publicznego. Zatem istotne jest, aby podstawowy schemat jednorazowego bezpiecznego podpisu Π umożliwiał podpisywanie wiadomości dłuższych niż klucz publiczny. Schemat Lamporta przedstawiony w rozdziale 12.6.1 nie ma tej właściwości. Jeśli jednak zastosujemy paradigmat hash-and-sign z sekcji 12.3 do schematu Lamporta, otrzymamy schemat jednorazowego bezpiecznego podpisu, który może podpisywać wiadomości o dowolnej długości. (Chociaż Twierdzenie 12.4 zostało sformułowane jedynie w odniesieniu do schematów podpisów spełniających definicję 12.2, nietrudno zauważać, że identyczny dowód działa w przypadku schematów podpisów jednorazowych.)

Ponieważ wynik ten jest kluczowy dla następnej części, stwierdzamy to formalnie. (Zauważ, że istnienie odpornych na kolizje funkcji skrótu implikuje istnienie funkcji jednokierunkowych; zobacz ćwiczenie 7.4.)

LEMMA 12.19 Jeżeli istnieją odporne na kolizje funkcje mieszające, to istnieje schemat podpisu jednorazowego (dla wiadomości o dowolnej długości).

Schemat podpisu opartego na łańcuchu to stanowy schemat podpisu, którego egzystencjalnie nie można podrobić w przypadku adaptacyjnego ataku z wybraną wiadomością. Ma jednak szereg wad. Po pierwsze, nie ma natychmiastowego sposobu na wyeliminowanie państwa (przypomnijmy, że naszym ostatecznym celem jest system bezpaństwowy spełniający Definicję 12.2). Nie jest to również zbyt wydajne, ponieważ długość podpisu, rozmiar stanu i czas weryfikacji są liniowe w stosunku do liczby podpisanych wiadomości. Wreszcie każdy podpis ujawnia wszystkie wcześniej podpisane wiadomości, co może być niepożądane w niektórych kontekstach.

12.6.3 Podpisy oparte na drzewie

Podpisującego w schemacie łańcuchowym z poprzedniej sekcji można postrzegać jako utrzymującego drzewo stopnia 1, zakończone w kluczu publicznym pk_1 i o głębokości równej liczbie podpisanych dotychczas wiadomości (por. rysunek 12.4). Naturalnym sposobem na poprawę wydajności jest zastosowanie drzewa binarnego, w którym każdy węzeł ma stopień 2. Tak jak poprzednio, podpis będzie odpowiadał „podpisanej” ścieżce w drzewie od liścia do korzenia; o ile drzewo ma głębokość wielomianową (nawet jeśli ma rozmiar wykładniczy!), weryfikację można przeprowadzić w czasie wielomianowym.

Konkretnie, aby podpisać wiadomości o długości n , będziemy pracować z drzewem binarnym o głębokości n posiadającym $2n$ liści. Tak jak poprzednio, sygnatarusz doda węzły do drzewa „w locie”, jeśli zajdzie taka potrzeba. Jednak w przeciwieństwie do schematu opartego na łańcuchu do podpisywania wiadomości będą używane tylko liście (a nie węzły wewnętrzne). Każdy liść drzewa będzie odpowiadał jednemu z możliwych komunikatów o długości n .

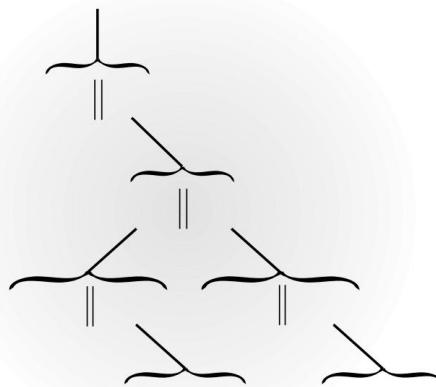
Bardziej szczegółowo, wyobrażamy sobie drzewo binarne o głębokości n , w którym pierwiastek jest oznaczony jako ϵ (tj. pusty ciąg znaków), a węzeł oznaczony ciągiem binarnym w (o długości mniejszej niż n) ma etykietę lewego dziecka w_0 i prawe dziecko oznaczone w_1 .

Drzewo to nigdy nie jest konstruowane w całości (należy pamiętać, że ma rozmiar wykładniczy), ale zamiast tego jest budowane przez osobę podpisującą w razie potrzeby.

Dla każdego węzła w kojarzymy parę kluczów pk_w , skw dla schematu jednorazowego bezpiecznego podpisu Π . Klucz publiczny korzenia, pk_ϵ , jest rzeczywistym kluczem publicznym osoby podpisującej. Aby podpisać wiadomość $m \in \{0, 1\}^n$, osoba podpisująca wykonuje następujące czynności:

1. Najpierw generuje klucze (w razie potrzeby) dla wszystkich węzłów na ścieżce od korzenia do liścia oznaczonego m . (Niektóre z tych kluczów publicznych mogły zostać wygenerowane w procesie podpisywania poprzednich wiadomości i w takim przypadku nie zostaną wygenerowane ponownie.)
2. Następnie „certyfikuje” ścieżkę od korzenia do liścia oznaczonego m , obliczając podpis na pk_w0pk_w1 , używając klucza prywatnego skw, dla każdego ciągu w będącego właściwym przedrostkiem m .
3. Na koniec „certyfikuje” siebie m , składając na m podpis przy użyciu klucza prywatnego skm.

Ostateczny podpis na m składa się z podpisu na m w odniesieniu do pk_m oraz wszystkich informacji potrzebnych do sprawdzenia ścieżki z liścia oznaczonego



RYSUNEK 12.5: Podpisy oparte na drzewie (koncepcyjnie).

m do korzenia; patrz rysunek 12.5. Dodatkowo podpisujący aktualizuje swój stan przechowując wszystkie klucze wygenerowane w ramach powyższego procesu podpisywania. Formalny opis tego schematu podano w rozdziale Konstrukcja 12.20.

Zauważ, że każdy z kluczowych kluczy w tym schemacie służy do podpisywania tylko jednej „wiadomości”. Każdy klucz powiązany z węzłem wewnętrznym podpisuje parę kluczy publicznych, a klucz na liściu służy do podpisywania tylko pojedynczej wiadomości. Ponieważ każdy klucz jest używany do podpisywania pary innych kluczy, ponownie potrzebujemy schematu jednorazowego bezpiecznego podpisu Π , aby móc podpisywać wiadomości dłuższe niż klucz publiczny. Lemat 12.19 pokazuje, że takie schematy można konstruować w oparciu o odpornie na kolizje funkcje mieszające.

Zanim udowodnisz bezpieczeństwo tego podejścia opartego na drzewach, zauważ, że pod wieloma względami poprawia ono schemat oparty na łańcuchu. Nadal pozwala na podpisywanie nieograniczonej liczby wiadomości. (Chociaż jest tylko $2n$ liści, przestrzeń wiadomości zawiera tylko $2n$ wiadomości. W każdym razie $2n$ jest ostatecznie większe niż jakakolwiek funkcja wielomianowa n .) Pod względem wydajności długość podpisu i czas weryfikacji są teraz proporcjonalne do długości wiadomości n , ale są niezależne od liczby podpisanych wiadomości. Schemat jest nadal stanowy, ale zobaczymy, jak można tego uniknąć, po udowodnieniu następującego wyniku.

TWIERDZENIE 12.21 Niech Π będzie schematem podpisu jednorazowego zabezpieczenia. Zatem Construction 12.20 jest schematem bezpiecznego podpisu.

DOWÓD Niech Π oznacza konstrukcję 12.20. Niech A będzie probabilistycznym wielomianem, przeciwnikiem czasu, niech (n) będzie górną granicą (wielomianu) $na = 2n \stackrel{\text{def}}{=} (n) + 1$.
liczba zapytań o podpisanie złożonych przez A , i ustawi $= (n)$

BUDOWA 12.20

Niech $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ będzie schematem podpisu. Dla ciągu binarnego m niech $m \in \{0, 1\}^n$, ϵ pusty ciąg. Skonstruuj schemat podpisu Π przy użyciu klucza publicznego pke i klucza prywatnego ske przedstawionego przez funkcję $\text{Sign}_{pke}(m)$ w następujący sposób:

- Gen : na wejściu $1n$ oblicz $(pke, ske) = \text{Gen}(1n)$ i wyprowadź klucz publiczny pke . Klucz prywatny i stan początkowy to ske .
- Znak ϵ : po wprowadzeniu wiadomości $m \in \{0, 1\}^n$, wykonaj następujące czynności.

1. Dla $i = 0$ do $n - 1$:

- Jeśli $pkm_{|i0}, pkm_{|i1} \in \sigma_m$ nie są w stanie, oblicz $(pkm_{|i0}, skm_{|i0}) = \text{Gen}(1n), (pkm_{|i1}, skm_{|i1}) = \text{Gen}(1n)$, oraz $\sigma_m \leftarrow \text{Sign}_{skm_{|i}}(pkm_{|i0})$ $pkm_{|i1}$. Ponadto dodaj wszystkie te wartości państwu.

2. Jeżeli σ_m nie jest jeszcze uwzględnione w stanie, oblicz $\sigma_m \leftarrow \text{Sign}_{skm}(m)$ i przechowywać go jako część stanu.

3. Wyprowadź podpis $\sigma_m \leftarrow pkm_{|0}, pkm_{|1}$

$\sum_{j=0}^{n-1} \sigma_m$.

- Vrfy : po wprowadzeniu klucza publicznego pke , wiadomości m i podpisu, wyjście $\sigma_m \leftarrow pkm_{|0}, pkm_{|1}$, $\sum_{j=0}^{n-1} \sigma_m = 1$ wtedy i tylko wtedy, gdy:

1. $\text{Vrfy}_{pkm}(pkm_{|0} \mid pkm_{|1}, \sigma_m) = 1$ dla wszystkich $i \in \{0, \dots, n-1\}$.

$\text{Vrfy}_{pkm}(m, \sigma_m) = 1$.

Schemat podpisu „oparty na drzewie”.

Należy pamiętać, że górną granicą liczby kluczy publicznych z Π potrzebnych do podpisów przy użyciu Π . Dzieje się tak dlatego, że każdy podpis w Π wymaga wygenerowania co najwyżej $2n$ nowych kluczy z Π (w najgorszym przypadku), a jeden dodatkowy klucz z Π jest używany jako rzeczywisty klucz publiczny pke .

Rozważmy następującego przeciwnika ppt A atakującego jednorazowe zabezpieczenie schematu natury Π :

Przeciwnik A: A

otrzymuje na wejściu klucz publiczny pk (parametr bezpieczeństwa n jest ukryty).

- Wybierz jednolity indeks i klawisze $\{1, \dots, n\}$. Skonstruuj listę pk_1, \dots, pk_n w następujący sposób:
 - Ustaw $pk_i := ok$.
 - Dla $i = i$ oblicz $(pk_i, sk_i) = \text{Gen}(1n)$.
- Uruchom A na wejściu klucza publicznego $pke = pk_1$. Kiedy A prosi o podpis w wiadomości, m:
 1. Dla $i = 0$ do $n - 1$:

- Jeżeli wartości $pkm|j0$, $pkm|j1$ oraz $\sigma_m|j$ nie zostały jeszcze zdefiniowane, to należy ustawić $pkm|j0$ i $pkm|j1$ na równe dwóm kolejnym nieużywanym kluczom publicznym pk_j i pk_{j+1} i obliczyć podpis $\sigma_m|j$ na $pkm|j0$ $pkm|j1$ w odniesieniu do $pkm|j$.
6

2. Jeżeli σ_m nie jest jeszcze zdefiniowane, oblicz sygnaturę σ_m na m względem do pkm (patrz przypis 6).

3. Podaj $\sigma_m|j$, $pkm|j0$, $pkm|j1$ σ_m do A .
 $n-1$
 $ja=0$,

- Powiedzmy, że A wysyła wiadomość m (dla której wcześniej nie prosi o podpis) i podpis $\sigma_m|j$, $pk_m|j0$, $pk_m|j1$. Jeśli jest to ważny podpis na m , to:

$$m|j1 \quad ja=0, \quad \sigma_m$$

Przypadek 1: Założmy, że istnieje $aj \in \{0, \dots, n-1\}$ dla którego $pk_m|j_0 = pkm|j_0$ lub $pk_m|j_1 = pkm|j_1$; obejmuję to przypadek, gdy $pkm|j_0$ lub $pkm|j_1$ nigdy nie zostały zdefiniowane przez A. Weźmy minimum takiego j i niech i będzie takie, że $pk_i = pkm|j = pk$ wyjście $m|j$ (takie i istnieje przez minimalność j). Jeśli $ja = ja$,
 $(pk_m|j_0, \quad pk_m|j_1, \quad \sigma_m|j)$.

Przypadek 2: Jeżeli przypadek 1 nie jest spełniony, to $pk_m = szt.$ Pozwól mi być takim $pk_i = pkm$. Jeśli $ja = ja$, moc wyjściowa (m, σ_m).

W eksperymencie $Sig-forge1-time, \Pi(n)$, widok A uruchamianego jako podprogram przez A rozkłada się identycznie jak widok A w eksperymencie $Sig-forgeA, \Pi(n)$.⁷ Zatem prawdopodobieństwo, że A wygeneruje fałszerstwo ma dokładnie $\delta(n)$, gdy jest uruchamiane jako podprogram przez A w tym eksperymencie. Biorąc pod uwagę, że A generuje fałszerstwo, rozważ każdy z dwóch możliwych przypadków opisanych powyżej:

Przypadek 1: jest jednolite i niezależne od poglądu A prawdopodobieństwo A zażądał podpisu. Ponieważ i to $i = j$ wynosi dokładnie 1/. Jeśli $ja = ja$, na = $pkm|j$ wiadomość $pkm|j0pkm|j1$ w odniesieniu do klucza publicznego $pk = pki$ została podana (i nie wymagała żadnych innych podpisów). Ponadto,

$$pk_m|j_0 \quad pk_m|j_1 = pkm|j_0pkm|j_1$$

a jednak o jest ważnym podpisem na pk pk w odniesieniu do pk . Zatem $m|j_0m|j_1$ A w tym przypadku generuje fałszerstwo.

Przypadek 2: Ponownie, został wybrany równomiernie losowo i jest niezależny od ponieważ i z punktu widzenia A prawdopodobieństwa, że $i = i$ wynosi dokładnie 1/. Jeśli $i = ja$, to A ,

⁶ Jeśli $i = j$, to A może samodzielnie obliczyć podpis w odniesieniu do pki . A może również uzyskać (pojedynczy) podpis w odniesieniu do pki , kierując odpowiednie zapytanie do swojej wyroczni podpisującej. To właśnie tutaj mamy na myśli.

⁷ Jak już wspomnieliśmy, kluczy publicznych A nigdy „nie zabraknie”. Zapytanie podpisujące A wykorzystuje 2n kluczy publicznych; zatem nawet gdyby do odpowiedzi na każde zapytanie A o podpisanie wymagane były nowe klucze publiczne (co generalnie nie będzie miało miejsca), A potrzebowałby tylko 2n (n) kluczy publicznych oprócz „głównego” klucza publicznego pke .

nie żądał żadnych podpisów w odniesieniu do klucza publicznego $pk = pki = pkm$ i jeszcze σ_m jest ważnym podpisem na m w odniesieniu do pk .

Widzimy, że pod warunkiem, że A wygeneruje fałszerstwo, A wygeneruje fałszerstwo z prawdopodobieństwem dokładnie $1/$. To znaczy że

$$\Pr[\text{Sig-forge}_{A, \Pi}(n) = 1] = \Pr[\text{Sig-forge}_{A, \Pi}(n) = 1]/(n).$$

Ponieważ Π jest schematem podpisu jednorazowego zabezpieczenia, istnieje dla niego pomijalna funkcja negl

$$\Pr[\text{Sig-forge}_{A, \Pi}(n) = 1] \leq \text{negl}(n).$$

Ponieważ jest to wielomian, oznacza to, że $\Pr[\text{Sig-forge}_{A, \Pi}(n) = 1]$ jest pomijalny. ■

Rozwiążanie bezpaństwowe

Jak opisano, osoba podpisująca generuje stan na bieżąco, w razie potrzeby. Możemy jednak wyobrazić sobie, że osoba podpisująca wygeneruje niezbędne informacje dla wszystkich węzłów w całym drzewie z wyprzedzeniem, w momencie generowania klucza. (Oznacza to, że w momencie generowania klucza osoba podpisująca mogłaby wygenerować klucze $\{(pkw, skw)\}$ i podpisy $\{\text{ow}\}$ dla wszystkich ciągów binarnych w o długości co najwyżej n .) Gdyby generowanie kluczy odbyło się w ten sposób, podpisujący nie musiałby w ogóle aktualizować swojego stanu; wszystkie te wartości mogłyby być przechowywane jako część (ogromnego) klucza prywatnego i otrzymaliśmy schemat bezstanowy. Problem z tym podejściem polega oczywiście na tym, że wygenerowanie wszystkich tych wartości wymagałoby wykładniczego czasu, a przechowywanie ich wszystkich wymagałoby pamięci wykładniczej.

Alternatywą jest przechowywanie pewnej losowości, która może zostać wykorzystana do wygenerowania wartości $\{(pkw, skw)\}$ i $\{\text{ow}\}$, w razie potrzeby, zamiast przechowywania samych wartości. Oznacza to, że podpisujący może przechowywać losowy串 rw dla każdego w i ilekroć potrzebne są wartości pkw, skw , podpisujący może obliczyć $(pkw, skw) := \text{Gen}(1n; rw)$, gdzie oznacza to wygenerowanie długości n klawisz używający losowych monet rw . Podobnie, jeśli procedura podpisywania jest probabilistyczna, podpisujący może, a następnie w ustawić $\text{ow} := \text{Signskw}(pkw0pkw1; r w)$ (zakładając, że sklep $r |w| < n$). Jednakże generowanie i przechowywanie wystarczającej liczby losowych ciągów znaków nadal wymaga wykładniczo czasu i pamięci.

Prosta modyfikacja tej alternatywy daje rozwiązanie w czasie wielomianowym. Zamiast zapisywać losowe rw i r , jak sugerowane powyżej, osoba podpisująca może przechowywać dwa klucze k , k dla funkcji pseudolosowej F . W razie potrzeby wartości pkw, skw można teraz wygenerować w następującym dwuetapowym procesie: 1.

Oblicz $rw := Fk(w)$.

2. Oblicz $(pkw, skw) := \text{Gen}(1n; rw)$ (jak poprzednio).

⁸Zakładamy, że długość wyjściowa F jest wystarczająco długa i że w jest dopełniane do pewnego ciągu o stałej długości w sposób jeden do jednego. Pomijamy tutaj te szczegóły techniczne.

Dodatkowo klucz k służy do generowania wartości r, która służy do obliczenia podpisu *ów*. Daje to schemat bezstanowy, w którym generowanie klucza (jak również podpisywanie i weryfikacja) może odbywać się w czasie wielomianowym. Intuicyjnie jest to bezpieczne, ponieważ przechowywanie funkcji losowej jest równoznaczne z przechowywaniem wszystkich potrzebnych wartości rw i r, a przechowywanie funkcji pseudolosowej jest „równie dobre”. Pozostawiamy to jako ćwiczenie mające na celu przedstawienie formalnego dowodu, że ten zmodyfikowany schemat pozostaje bezpieczny.

Ponieważ istnienie odpornych na kolizje funkcji skrótu implikuje istnienie funkcji jednokierunkowych (por. ćwiczenie 7.4), a ta ostatnia implikuje istnienie funkcji pseudolosowych (patrz rozdział 7), mamy:

TWIERDZENIE 12.22 Jeśli istnieją odporne na kolizje funkcje mieszające, to istnieje (bezstanowy) schemat bezpiecznego podpisu.

Zauważmy, że możliwe jest skonstruowanie schematów sygnatur spełniających definicję 12.2 przy (minimalnym) założeniu, że istnieją funkcje jednokierunkowe; dowód tego wyniku wykracza poza zakres tej książki.

12.7 *Certyfikaty i infrastruktura klucza publicznego

W tej sekcji pokrótko omawiamy jedno z głównych zastosowań podpisów cyfrowych: bezpieczną dystrybucję kluczy publicznych. To zatacza koło w naszej dyskusji na temat kryptografii klucza publicznego. W tym i poprzednim rozdziale widzieliśmy, jak używać kryptografii klucza publicznego po bezpiecznej dystrybucji kluczy publicznych. Teraz pokażemy, jak samą kryptografię klucza publicznego można wykorzystać do bezpiecznej dystrybucji kluczy publicznych. Może się to wydawać okrężne, ale tak nie jest. Pokażemy, że gdy pojedynczy klucz publiczny, należący do zaufanej strony, zostanie rozdystrybuowany w bezpieczny sposób, może on zostać użyty do „bootstrapa” bezpiecznej dystrybucji dowolnie wielu innych kluczy publicznych. Zatem przynajmniej w zasadzie problem bezpiecznej dystrybucji kluczy należy rozwiązać tylko raz.

Kluczowym pojęciem jest tutaj certyfikat cyfrowy, będący po prostu podpisem wiążącym podmiot z jakimś kluczem publicznym. Mówiąc konkretnie, powiedzmy, że Charlie wygenerował parę kluczy (pkC , skC) dla bezpiecznego schematu podpisu cyfrowego (w tej sekcji będziemy zajmować się jedynie schematami podpisów spełniającymi definicję 12.2). Założmy dalej, że inna strona Bob również wygenerowała parę kluczy (pkB , skB) (w tej dyskusji mogą to być klucze albo do schematu podpisu, albo do schematu szyfrowania z kluczem publicznym) i że Charlie wie, że pkB jest kluczem Boba klucz publiczny. Wtedy Charlie może obliczyć podpis

$$\text{certifikatC} \stackrel{\text{def}}{=} \text{Sign}_{\text{skC}}(\text{„Klucz Boba to pkB”})$$

i przekaź ten podpis Bobowi. CertC = B nazywamy certyfikatem klucza Boba

wydany przez Charliego. W praktyce certyfikat powinien jednoznacznie identyfikować stronę posiadającą konkretny klucz publiczny, dlatego można zastosować bardziej jednoznaczne określenie opisowe niż „Bob”, na przykład imię i nazwisko Boba oraz adres e-mail lub adres URL strony internetowej Boba.

Powiedzmy teraz, że Bob chce się porozumieć z inną osobą, Alicją, która już zna pkC. Bob może wysłać (pkB, certC → B) do Alicji, która może następnie sprawdzić, czy certC → B jest rzeczywiście prawidłowym podpisem w wiadomości „Kluczem Boba jest pkB” w odniesieniu do pkC. Zakładając, że weryfikacja się powiedzie, Alicja wie, że Charlie podpisał wskazaną wiadomość. Jeśli Alicja ufa Charliemu, może zaakceptować pkB jako prawidłowy klucz publiczny Boba.

Wszelka komunikacja między Bobem i Alicją może odbywać się w sposób niepewny i niepewny nieuwierzytelny kanał. Jeśli aktywny przeciwnik zakłóca transmisję (pkB, certC → B) od Boba do Alicji, przeciwnik ten nie będzie w stanie wygenerować ważnego certyfikatu łączącego Boba z jakimkolwiek innym kluczem publicznym pk, chyba że Charlie wcześniej podpisał jakiś inny certyfikat łączący Bob z p

(w takim przypadku i tak nie jest to duży atak). Wszystko to zakłada, że Charlie nie jest nieuczciwy i że jego klucz prywatny nie został naruszony.

W powyższym opisie pominęliśmy wiele szczegółów. Co najważniejsze, nie omawialiśmy w ogóle tego, jak Alicja uczy się pkC; jak Charlie może być pewien, że pkB jest kluczem publicznym Boba; i jak Alicja decyduje, czy zaufać Charliemu. Pełne określenie takich szczegółów (i innych) definiuje infrastrukturę klucza publicznego (PKI), która umożliwia szeroką dystrybucję kluczy publicznych.

Zaproponowano wiele różnych modeli PKI, a teraz wymienimy kilka z bardziej popularnych. Nasze traktowanie tutaj będzie utrzymane na stosunkowo wysokim poziomie, a czytelnikowi zainteresowanemu dalszymi szczegółami zalecamy zapoznanie się z literaturą na końcu tego rozdziału.

Pojedynczy urząd certyfikacji. Najprostsza infrastruktura PKI zakłada istnienie jednego urzędu certyfikacji (CA), któremu wszyscy ufają i który wydaje certyfikaty dla każdego klucza publicznego. Organem certyfikującym nie byłaby zazwyczaj osoba, ale raczej firma, której zadaniem jest certyfikowanie kluczy publicznych, agencja rządowa lub być może dział w organizacji (chociaż w tym drugim przypadku urząd certyfikacji prawdopodobnie być używane przez osoby w organizacji). Każdy, kto chce polegać na usługach urzędu certyfikacji, musiałby uzyskać legalną kopię klucza publicznego pkCA urzędu certyfikacji.

Jest oczywiste, że ten krok musi zostać przeprowadzony w sposób bezpieczny, ponieważ jeśli jakaś strona uzyska niepoprawną wersję pkCA, wówczas ta strona może nie być w stanie uzyskać autentycznej kopii klucza publicznego innej osoby. Oznacza to, że pkCA musi być dystrybuowane za pośrednictwem uwierzytelionego kanału. Najprościej można to zrobić za pomocą środków fizycznych: na przykład, jeśli urząd certyfikacji znajduje się w organizacji, każdy pracownik może uzyskać autentyczną kopię pkCA bezpośrednio od urzędu certyfikacji pierwszego dnia pracy. Jeśli urząd certyfikacji jest firmą, inni użytkownicy będą musieli w pewnym momencie udać się do tej firmy i, powiedzmy, wziąć pamięć USB zawierającą klucz publiczny urzędu certyfikacji. Tę niewygodną czynność należy wykonać tylko raz.

W praktyce powszechnym sposobem dystrybucji klucza publicznego przez urząd certyfikacji jest „bu-

dle" ten klucz publiczny za pomocą innego oprogramowania. Dzieje się tak na przykład obecnie w wielu popularnych przeglądarkach internetowych: klucz publiczny urzędu certyfikacji jest dostarczany razem z przeglądarką, a przeglądarka jest zaprogramowana tak, aby automatycznie weryfikowała certyfikaty po ich otrzymaniu. (W rzeczywistości nowoczesne przeglądarki internetowe mają klucze publiczne wielu urzędów certyfikacji wbudowane w swój kod, dlatego dokładniej mówiąc są w omówionym poniżej modelu „wielu urzędów certyfikacji”).

Mechanizm, za pomocą którego urząd certyfikacji wystawia certyfikat jakiejś stronie Bob, również musi być bardzo dokładnie kontrolowany, chociaż szczegóły mogą się różnić w zależności od urzędu certyfikacji. Na przykład Bob może być zmuszony do osobistego stawienia się z kopią swojego klucza publicznego pkB wraz z dokumentem tożsamości potwierdzającym, że jego imię i nazwisko (lub adres e-mail) jest tym, co twierdzi. Dopiero wtedy urząd certyfikacji wystawi certyfikat.

W modelu, w którym istnieje jeden urząd certyfikacji, strony całkowicie ufają temu urzędowi, że będzie wystawiać certyfikaty tylko w stosownych przypadkach; dlatego tak ważne jest, aby przed wydaniem certyfikatu przeprowadzić szczegółowy proces weryfikacji. W konsekwencji, jeśli Alicja otrzyma certyfikat certCA – B poświadczający, że pkB jest kluczem publicznym Boba, Alicja zaakceptuje to stwierdzenie jako ważne i użyje pkB jako klucza publicznego Boba.

Wiele urzędów certyfikacji. O ile model, w którym istnieje tylko jeden urząd certyfikacji, jest prosty i atrakcyjny, o tyle nie jest zbyt praktyczny. Po pierwsze, poza jedną organizacją jest mało prawdopodobne, aby wszyscy ufali temu samemu urzędowi certyfikacji. Nie musi to oznaczać, że ktoś uważa, że urząd certyfikacji jest skorumpowany; może się po prostu zdarzyć, że ktoś uzna proces weryfikacji przeprowadzany przez urząd certyfikacji za niewystarczający (powiedzmy, że urząd certyfikacji prosi o tylko jedną formę identyfikacji podczas generowania certyfikatu, ale Alicja wolałaby, aby zamiast tego użyto dwóch). Co więcej, urząd certyfikacji jest pojedynczym punktem awarii całego systemu. Jeżeli urząd certyfikacji jest skorumpowany, można go przekupić, lub nawet jeśli urząd certyfikacji po prostu niedbale chroni swój klucz prywatny, legalność wydawanych certyfikatów może zostać zakwestionowana. Jest to również niewygodne dla wszystkich stron, które chcą certyfikatów, aby musiały kontaktować się z tym urzędem certyfikacji.

Jednym ze sposobów rozwiązyania tych problemów jest poleganie na wielu urzędach certyfikacji. Strona Bob, która chce uzyskać certyfikat na swoim kluczu publicznym, może wybrać, które urzędy certyfikacji chce wystawić certyfikat, a strona Alicja, której zostanie przedstawiony certyfikat lub nawet wiele certyfikatów wydanych przez różne urzędy certyfikacji, może wybrać, który z nich Certyfikaty CA, którym ufa. Nie ma nic złego w uzyskaniu przez Boba certyfikatu od więcej niż jednego urzędu certyfikacji (poza pewnymi niedogodnościami i kosztami dla Boba), ale Alicja musi zachować większą ostrożność, ponieważ bezpieczeństwo jej komunikacji jest ostatecznie tak dobre, jak najmniej bezpieczny urząd certyfikacji, który ona ufa. To znaczy, powiedzmy, że Alicja ufa dwóm urzędom certyfikacji CA1 i CA2, a urząd CA2 jest uszkodzony przez przeciwnika. Wtedy, chociaż przeciwnik ten nie będzie mógł sfałszować certyfikatów wydanych przez CA1, będzie mógł wystawiać fałszywe certyfikaty w imieniu CA2 dla dowolnej wybranej przez siebie tożsamości/klucza publicznego. Jest to prawdziwy problem w obecnych systemach. Jak wspomniano wcześniej, systemy operacyjne/przeglądarki internetowe są zazwyczaj dostarczane z wstępnie skonfigurowanymi kluczami publicznymi wielu urzędów certyfikacji, a ustaleniem domyślnym jest traktowanie wszystkich tych urzędów certyfikacji jako jednakowo godnych zaufania. Zasadniczo jednak każda firma, która jest skłonna zapłacić, może zostać uwzględniona w roli urzędu certyfikacji. Zatem lista wstępnie skonfigurowanych urzędów certyfikacji obejmuje kilka renomowanych firm o ugruntowanej pozycji

wraz z innymi, nowszymi firmami, których wiarygodności nie da się łatwo ustalić. Użytkownikowi pozostaje ręczne skonfigurowanie ustawień tak, aby akceptował tylko certyfikaty z urzędów certyfikacji, którym użytkownik ufa.

Łańcuchy delegacji i certyfikatów. Innym podejściem, które zmniejsza część obciążenia pojedynczego urzędu certyfikacji (ale nie rozwiązuje problemów związanych z bezpieczeństwem wynikającym z posiadania pojedynczego punktu awarii), jest użycie łańcuchów certyfikatów. Przedstawiamy pomysł na łańcuchy certyfikatów o długości 2, chociaż łatwo zauważać, że wszystko, co mówimy, uogólnia się na łańcuchy o dowolnej długości.

Załóżmy, że Charlie, występujący w charakterze urzędu certyfikacji, wystawia Bobowi certyfikat, tak jak w naszej pierwotnej dyskusji. Założymy dalej, że klucz pkB Boba jest kluczem publicznym schematu podpisu. Bob z kolei może wystawiać własne certyfikaty dla innych podmiotów. Na przykład Bob może wystawić certyfikat Alicji w formie

$$\text{certifikatB} \stackrel{\text{def}}{=} \text{SignskB}(\text{,,Klucz Alicji to pkA"}).$$

Jeśli Alice chce się porozumieć z jakąś czwartą osobą, Dave'em, kto wie Klucz publiczny Chariego (ale nie Boba), wtedy Alicja może wysłać

$$\text{pkA, certB A, pkB, certC B,}$$

do Dave'a. Co Dave może z tego wywnioskować? Cóż, może najpierw sprawdzić, czy Charlie, któremu ufa i którego klucz publiczny jest już w jego posiadaniu, podpisał certyfikat certC B wskazujący, że pkB rzeczywiście należy do kogoś o imieniu Bob. Dave może również sprawdzić, czy osoba o imieniu Bob podpisała certyfikat certB A wskazujący, że pkA rzeczywiście należy do Alicji. Jeśli Dave ufa Charliemu, że będzie wystawiał certyfikaty tylko osobom godnym zaufania, wówczas Dave może zaakceptować pkA jako autentyczny klucz Alicji.

Podkreślamy, że w tym przykładzie silniejsza semantyka jest powiązana z certyfikatem certC B. W naszej wcześniejszej dyskusji certyfikat w tym formularzu był jedynie potwierdzeniem, że Bob posiada klucz publiczny pkB. Teraz certyfikat potwierdza, że Bob posiada klucz publiczny pkB i Bob może wystawiać inne certyfikaty. Kiedy Charlie podpisuje certyfikat dla Boba, który ma silniejszą semantykę, w efekcie Charlie przekazuje Bobowi swoje uprawnienia do wydawania certyfikatów. Bob może teraz działać jako pełnomocnik Chariego i wystawiać certyfikaty w jego imieniu.

Wracając do infrastruktury PKI opartej na urzędzie certyfikacji, możemy sobie wyobrazić jeden „główny” urząd certyfikacji i n „drugiego poziomu” urzędów certyfikacji CA1, . . . , Mów. Główny urząd certyfikacji może wystawiać certyfikaty dla każdego z urzędów certyfikacji drugiego poziomu, który z kolei może wystawiać certyfikaty dla innych zasad posiadających klucze publiczne. Odciąża to główny urząd certyfikacji, a także ułatwia stronom uzyskiwanie certyfikatów (ponieważ mogą teraz na przykład kontaktować się z najbliższym urzędem certyfikacji drugiego poziomu). Z drugiej strony zarządzanie tymi urzędami certyfikacji drugiego poziomu może być trudne, a ich obecność oznacza, że w systemie jest teraz więcej punktów ataku.

Model „sieci zaufania”. Ostatnim przykładem infrastruktury PKI, który omówimy, jest model w pełni rozproszony, bez centralnych punktów zaufania, zwany „siecią

zaufanie." Odmiana tego modelu jest wykorzystywana przez oprogramowanie PGP („Pretty Good Privacy") do szyfrowania wiadomości e-mail w celu dystrybucji kluczy publicznych.

W modelu „sieci zaufania” każdy może wystawiać certyfikaty komukolwiek innemu, a każdy użytkownik musi sam podjąć decyzję o tym, jak bardzo ufa certyfikatom wydanym przez innych użytkowników. Jako przykład tego, jak to może działać, założmy, że użytkownik Alicja jest już w posiadaniu kluczy publicznych pk1, pk2, pk3 dla niektórych użytkowników C1, C2, C3. (Poniżej omawiamy, w jaki sposób Alicja może początkowo uzyskać te klucze publiczne.) Inny użytkownik Bob, który chce komunikować się z Alicją, może posiadać certyfikaty certC1 B, certC3 B i certC4 B i wyśle te certyfikaty (wraz ze swoim klucz publiczny pkB) do Alicji. Alicja nie może zweryfikować certyfikatu C4 B (ponieważ nie ma klucza publicznego C4), ale może zweryfikować pozostałe dwa certyfikaty. Teraz musi zdecydować, jak duże zaufanie pokłada w C1 i C3. Może zdecydować się na przyjęcie pkB, jeśli jednoznacznie ufa C1 lub w mniejszym stopniu ufa zarówno C1 , jak i C3 . (Może na przykład uznać za prawdopodobne, że albo C1 , albo C3 jest uszkodzony, ale uważać za mało prawdopodobne, aby oba były uszkodzone.)

W tym modelu, jak opisano, od użytkowników oczekuje się gromadzenia zarówno kluczy publicznych innych podmiotów, jak i certyfikatów na własnym kluczu publicznym. W kontekście PGP zwykle robiono to na „stronach podpisujących klucze”, podczas których użytkownicy PGP spotykali się (powiedzmy na konferencji), przekazywali sobie nawzajem autentyczne kopie swoich kluczy publicznych i wystawiali sobie nawzajem certyfikaty. Ogólnie rzeczą biorąc, użytkownicy strony podpisującej klucze mogą się nie znać, ale mogą na przykład sprawdzić prawo jazdy przed przyjęciem lub wydaniem certyfikatu czynnego klucza publicznego.

Klucze publiczne i certyfikaty mogą być również przechowywane w centralnej bazie danych i dzieje się tak w przypadku PGP (patrz <http://pgp.mit.edu>). Kiedy Alicja chce wysłać zaszyfrowaną wiadomość do Boba, może wyszukać klucz publiczny Boba w tej bazie danych; wraz z kluczem publicznym Boba baza danych zwróci listę wszystkich posiadanych certyfikatów, które zostały wydane dla klucza publicznego Boba. Możliwe jest również, że w bazie danych znajdzie się wiele kluczy publicznych Boba, a każdy z tych kluczy publicznych może być poświadczony certyfikatami wydanymi przez inny zestaw stron. Po raz kolejny Alicja musi zdecydować, jak bardzo zaufać któremukolwiek z tych kluczy publicznych, zanim ich użyje.

Model sieci zaufania jest atrakcyjny, ponieważ nie wymaga zaufania do żadnego organu centralnego. Z drugiej strony, chociaż może to działać dobrze w przypadku przeciętnego użytkownika szyfrującego swoją pocztę e-mail, nie wydaje się odpowiednie w przypadku ustawiń, w których bezpieczeństwo jest bardziej krytyczne, lub w przypadku dystrybucji organizacyjnych kluczy publicznych (np. w przypadku handlu elektronicznego w Internecie).. Jeśli użytkownik chce na przykład komunikować się ze swoim bankiem, jest mało prawdopodobne, aby zaufał osobom spotkanym na konferencji w zakresie certyfikacji klucza publicznego jego banku, a także jest mało prawdopodobne, aby przedstawiciel banku udał się do strony podpisującej klucze, aby uzyskać certyfikowany klucz banku.

Unieważnianie certyfikatów

Ważną kwestią, której jeszcze w ogóle nie poruszyliśmy, jest fakt, że certyfikaty zasadniczo nie powinny być ważne bezterminowo. Pracownik może odejść

firma – w takim przypadku osoba ta nie może już otrzymywać zaszyfrowanej komunikacji od innych osób w firmie; klucz prywatny użytkownika może również zostać skradziony, w którym to momencie użytkownik (zakładając, że wie o kradzieży) będzie chciał wygenerować nową parę kluczy i usunąć stary klucz publiczny z obiegu. W każdym z tych scenariuszy potrzebujemy sposobu na unieważnienie wcześniejszych wydanych certyfikatów.

Podejścia do rozwiązywania tych problemów są różnorodne i złożone, dlatego wspomnimy tylko o dwóch stosunkowo prostych pomysłach, które w pewnym sensie reprezentują przeciwnie skrajności. (Udoskonalanie tych metod jest aktywnym obszarem badań nad bezpieczeństwem sieci w świecie rzeczywistym.)

Wygaśnięcie. Jedną z metod zapobiegania nieograniczonemu używaniu certyfikatów jest umieszczenie w certyfikacie daty wygaśnięcia. Certyfikat wydany przez urząd certyfikacji Charliego dla klucza publicznego Boba może teraz mieć tę formę

$$\text{certyfikatC} \stackrel{\text{def}}{=} \text{SignskC}(\text{„Klucz Boba to pkB”, data}),$$

gdzie data oznacza datę w przeszłości, w którym to momencie certyfikat traci ważność. (Przykładowo rok od dnia wystawienia certyfikatu.) Kiedy inny użytkownik będzie weryfikował ten certyfikat, musi znać nie tylko pkB, ale i datę ważności, a teraz musi sprawdzić nie tylko, czy podpis jest ważny, ale również, że nie minął termin ważności. Użytkownik posiadający certyfikat musi skontaktować się z urzędem certyfikacji w celu uzyskania nowego certyfikatu po wygaśnięciu aktualnego certyfikatu; w tym momencie urząd certyfikacji ponownie weryfikuje tożsamość/poświadczanie użytkownika przed wydaniem kolejnego certyfikatu.

Stosowanie dat ważności zapewnia bardzo ogólne rozwiązanie problemów wspomnianych wcześniej. Jeżeli pracownik odchodzi z firmy następnego dnia po otrzymaniu certyfikatu, a certyfikat traci ważność po roku od daty jego wydania, to pracownik ten może bezprawnie posługiwać się swoim kluczem publicznym przez cały rok, aż do upływu terminu ważności. Z tego powodu to podejście jest zwykle stosowane w połączeniu z innymi metodami, takimi jak ta, którą opiszemy poniżej.

Unieważnienie. Gdy pracownik odchodzi z organizacji lub klucz prywatny użytkownika zostaje skradziony, chceliśmy, aby certyfikaty wydane dla jego kluczy publicznych utraciły ważność natychmiast, a przynajmniej tak szybko, jak to możliwe.

Można to osiągnąć poprzez wyraźne unieważnienie certyfikatu przez urząd certyfikacji. Dla uproszczenia zakładamy, że istnieje jeden urząd certyfikacji, ale wszystko, co mówimy, ma zastosowanie bardziej ogólnie, jeśli użytkownik posiada certyfikaty wystawione przez wiele urzędów certyfikacji.

Istnieje wiele różnych sposobów obsługi odwołania. Jedna z możliwości (jedyna, którą omówimy) polega na tym, że urząd certyfikacji umieszcza numer seryjny na każdym wystawianym przez siebie certyfikacie; oznacza to, że certyfikat będzie teraz miał formę

$$\text{certyfikatC} \stackrel{\text{def}}{=} \text{SignskC}(\text{„Klucz Boba to pkB”, ###}),$$

gdzie „###” oznacza numer seryjny tego certyfikatu. Każdy certyfikat powinien mieć unikalny numer seryjny, a urząd certyfikacji będzie przechowywać informacje (Bob, pkB, ###) dla każdego wygenerowanego przez siebie certyfikatu.

Jeśli klucz prywatny użytkownika Bob odpowiadający kluczowi publicznemu pkB zostanie skradziony, Bob może powiadomić o tym fakcie urząd certyfikacji. (Urząd certyfikacji musi tutaj zweryfikować tożsamość Boba, aby uniemożliwić innemu użytkownikowi fałszywe unieważnienie certyfikatu wydanego Bobowi.) Urząd certyfikacji przeszuka następnie swoją bazę danych w celu znalezienia numeru seryjnego powiązanego z certyfikatem wydanym dla Boba i pkB. Na przykład pod koniec każdego dnia urząd certyfikacji wygeneruje listę odwołań certyfikatów (CRL) zawierającą numery seryjne wszystkich unieważnionych certyfikatów i podpisze listę CRL oraz bieżącą datę.

Podpisana lista CRL jest następnie szeroko rozpowszechniana lub w inny sposób udostępniana potencjalnym weryfikatorom. Weryfikacja certyfikatu wymaga obecnie sprawdzenia, czy podpis w certyfikacie jest ważny, sprawdzenia, czy numer seryjny nie pojawia się na najbardziej aktualnej liście unieważnień oraz sprawdzenia podpisu urzędu certyfikacji na samej liście unieważnień.

W tym podejściu, tak jak to opisaliśmy, istnieje przerwa maksymalnie jednego dnia, zanim certyfikat stanie się nieważny. Zapewnia to większą elastyczność niż podejście oparte wyłącznie na datach wygaśnięcia.

12.8 Łączenie wszystkiego w jedną całość – SSL/TLS

W kulminacji tego, co omówiliśmy w tej książce, omówimy protokół Transport Layer Security (TLS), który jest powszechnie używany do zabezpieczania komunikacji w Internecie; TLS to protokół używany przez Twoją przeglądarkę za każdym razem, gdy łączysz się z witryną internetową za pomocą protokołu https , a nie http. Powinno być jasne, że naszym celem jest skupienie się na podstawowej kryptografii stosowanej w podstawowym protokole TLS, a nie na różnych innych aspektach, które choć są interesujące z punktu widzenia bezpieczeństwa sieci, są nieistotne dla naszych obaw. Jak zwykle nieco uprościliśmy i wyabstrahowaliśmy części protokołu, aby przekazać główny punkt i nie należy polegać na naszym opisie przy implementacji. Wreszcie, formalnie nie definiujemy ani nie twierdzimy, że protokół jest bezpieczny; w istocie formalna analiza TLS jest przedmiotem aktywnych badań.

TLS to ustandaryzowany protokół oparty na prekursorze zwanym SSL (lub Secure Socket Layer), opracowanym przez firmę Netscape w połowie lat 90-tych; ostatnią dostępną wersją był SSL 3.0. TLS w wersji 1.0 został wydany w 1999 r., zaktualizowany do wersji 1.1 w 2006 r. i ponownie zaktualizowany do wersji 1.2 (aktualna wersja) w 2008 r. W chwili pisania tego tekstu około 50% witryn internetowych nadal korzysta z protokołu TLS 1.0, a nie z bardziej najnowsza wersja; wszystkie główne przeglądarki internetowe obsługują TLS 1.2, chociaż w niektórych przypadkach domyślnie używane są wcześniejsze wersje TLS.

W przeważającej części nasz poniższy opis jest na tyle wysokim poziomie, że różnice pomiędzy wersjami są dla naszych celów nieistotne. Ostrzegamy jednak, że istnieje kilka znanych ataków na wcześniejsze wersje.

TLS umożliwia klientowi (np. przeglądarce internetowej) i serwerowi (np. stronie internetowej) uzgodnienie zestawu wspólnych kluczy, a następnie wykorzystanie tych kluczy do szyfrowania i

uwierzytelniać ich późniejszą komunikację. Składa się z dwóch części: protokołu uzgadniania, który przeprowadza uwierzytelioną wymianę kluczy w celu ustalenia wspólnych kluczy, oraz protokołu warstwy rekordu, który wykorzystuje te wspólne klucze do szyfrowania/uwierzytelniania komunikacji stron. Chociaż protokół TLS umożliwia klientom uwierzytelnianie na serwerach, jest on używany głównie do uwierzytelniania serwerów na rzecz klientów, ponieważ zazwyczaj tylko serwery posiadają certyfikaty. (Po ustanowieniu sesji TLS uwierzytelnianie między użytkownikiem a serwerem —jeśli jest to wymagane —można przeprowadzić w warstwie aplikacji stosu sieciowego, np. wysyłając hasło).

Protokół uścisku dłoni. Opiszemy teraz podstawowy przebieg protokołu uzgadniania. Na początku protokołu klient C przechowuje zestaw kluczy publicznych urzędu certyfikacji $\{pk_1, \dots, pk_n\}$, a serwer S ma parę kluczy (pk_S, sk_S) dla KEM9 wraz z certyfikatem $certi_S$ wydanym przez jeden z urzędów certyfikacji, którego klucz publiczny C zna. Aby połączyć się z S, strony wykonują następujące kroki.

1. C rozpoczyna od wysłania wiadomości do S, która zawiera informacje o wersjach protokołu obsługiwanej przez klienta, mechanizmach szyfrowania obsługiwanych przez klienta (np. jakie funkcje mieszające lub szyfry blokowe pozwalają na to klient) oraz jednorodną wartość („nonce”) NC.
2. S odpowiada, wybierając najnowszą wersję obsługiwanej protokołu oraz odpowiedni zestaw szyfrów. Dodatkowo wysyła swój klucz publiczny pk_S , swój certyfikat $certi_S$, i własną jednorodną wartość NS.
3. C sprawdza, czy jeden z posiadanych kluczy publicznych urzędu certyfikacji, np. pki , pasuje do urzędu certyfikacji, który wydał certyfikat S. Jeśli tak, C weryfikuje certyfikat (a także sprawdza, czy nie wygasł lub nie został unieważniony) i, jeśli się powiedzie, dowiaduje się, że pk_S jest kluczem publicznym S. Następnie uruchamia (c, pmk) EncapspkS(1n) (patrz sekcja 11.3), aby uzyskać zaszyfrowany tekst c i tak zwany klucz przedgławny pmk . Wysyła c na serwer.
 pmk służy do wyprowadzenia klucza głównego mk przy użyciu funkcji wyprowadzania klucza (por. rozdział 5.6.4) zastosowanej do pmk , NC i NS. Następnie klient stosuje kS generator pseudolosowy do mk w celu uzyskania czterech kluczy $kC, k_{C'}, k_{S'}, k_{S''}$.
 Na koniec C oblicza $\tau_C = Mac_{mk}(transkrypcja)$, gdzie transkrypcja oznacza wszystkie wiadomości wymienione dotychczas pomiędzy C i S. Następnie klient wysyła τ_C do S. (W rzeczywistości τ_C jest samo w sobie szyfrowane i uwierzytelniane, tak jak ma to miejsce w przypadku komunikacji w warstwie rekordu, co opisano poniżej).
4. S oblicza $pmk := DecapsskS(c)$, z którego może wyprowadzić mk i kC , tak jak zrobił to klient. Gdy $k_{C'} \neq k_{C''}$ (transkrypcja, τ_C) = 1, wówczas S przerwa działanie. W przeciwnym razie ustawia $\tau_S = Mac_{mk}(transkrypt)$, gdzie transkrypcja oznacza wszystkie wiadomości wymienione dotychczas pomiędzy C i S (tzn. włączając

⁹ Możliwe jest również, że S będzie posiadał klucz publiczny dla schematu podpisu i używał efemerycznego klucza publicznego KEM w poniższym protokole. Nie omawiamy tej bardziej skomplikowanej opcji.

ostatnia wiadomość otrzymana od C). Następnie S wysyła τ S do C. (Ponownie τ S wynosi faktycznie szyfrowany i uwierzytelniany tak, jak ruch w warstwie rekordów.)

5. Jeśli $Vrfymk(transkrypt, \tau S) = 1$, klient przerwa.

Po pomyślnym wykonaniu protokołu uzgadniania C i S dzielą się zestawem czterech kluczy symetrycznych $k_C, k_{C'}, k_S, k_{S'}$.

TLS 1.2 obsługuje dwa KEM: KEM oparty na CDH/DDH, jak w Construction 11.19, lub schemat szyfrowania oparty na RSA PKCS #1 v1.5. Aby zapobiec

Ataki w stylu Bleichenbachera na schemat w tym drugim przypadku serwer powinien nie zgłaszać błędu deszyfrowania, jeśli Decaps nie powiedzie się w kroku 4. Zamiast tego, jeśli odszyfrowanie zawiedzie, powinien wybrać jednolity pmk, a następnie kontynuować działanie protokołu używając tej wartości. (W takim wypadku oczywiście wartość τ C nie zostanie zweryfikowana i S przerwie; chodzi o to, że atakujący nie jest w stanie rozróżnić, czy tak było z powodu niepowodzenia deszyfrowania, czy nie.)

Intuicja dotycząca bezpieczeństwa protokołu uzgadniania jest taka, że ponieważ C weryfikuje certyfikat, wie, że tylko legalny serwer S może uczyć się PMK i stąd mk. Zatem, jeśli protokół zakończy się pomyślnie, C o tym wie dzieli klucze k_C i k_S z legalnym kierzącym przeciwnikiem, który nie nauczył się żadnego klucza. Powinno to obowiązywać nawet w obecności aktywnego atakującego, choć formalny dowód na to jest dość skomplikowany. Zauważamy, jednak do tego przyzwyczajony jest kod uwierzytelniający wiadomość znajdującej się w transkrypcji uniemożliwić atakującemu typu „man-in-the-middle” zmianę wersji protokołu i zestawy szyfrów wysypane na początku protokołu uzgadniania. To zapobiega uniemożliwieniu atakującemu spowodowania użycia przez S lub C starych, słabszych wersji protokołu, lub słabego blokowego i krótkiego klucza.

Protokół warstwy rekordu. Po uzgodnieniu kluczy przez C i S, strony używają tych kluczy do szyfrowania i uwierzytelniania wszystkich kolejnych transakcji Komunikacja. C wykorzystuje k_C (odpowiednio $k_{C'}$) do szyfrowania (odpowiednio uwierzytelniania) wszystkich mówców wysyła do S; podobnie S używa k_S i kierzącego S wiadomości - do szyfrowania i uwierzytelniania wszystkich wiadomości, które wysyła. Numery sekwencyjne służą do zapobiegania atakom polegającym na powtarzaniu, jak omówiono w sekcji 4.5.3. TLS 1.2 wykorzystuje metodę uwierzytelniania, a następnie szyfrowania, co, jak widzieliśmy w sekcji 4.5.2, może być problematyczne.

12.9 *Szyfrowanie podpisu

Aby zamknąć ten rozdział, krótko i nieformalnie omówimy kwestię wspólnych poufności i integralności w ustawieniach klucza publicznego. Chociaż jest to zbieżne z naszym traktowaniem z sekcji 4.5, faktem, że jesteśmy teraz w ustawieniu klucza publicznego wprowadza kilka dodatkowych komplikacji.

Dla uproszczenia rozważamy sieć, w której wszystkie zainteresowane strony mają pary kluczy publiczny/prywatny zarówno do szyfrowania, jak i podpisywania. Niech (ek, dk) oznacza

(publiczny) klucz szyfrowania i (prywatny) klucz odszyfrowywania oraz użyj (vk, sk) jako (publicznego) klucza weryfikacyjnego i (prywatnego) klucza podpisywania. Zakładamy, że wszystkie strony znają klucze publiczne innych osób.

Nieformalnie naszym celem jest zaprojektowanie mechanizmu, który pozwoli nadawcy wysłać wiadomość m do odbiorcy R, zapewniając jednocześnie, że (1) żadna inna strona w sieci nie będzie mogła poznać żadnych informacji o m (tj. tajemnicy) oraz (2) R ma pewność, że wiadomość pochodzi od S (tzn. integralność). Będziemy chcieli wziąć pod uwagę obie te właściwości zabezpieczeń nawet przed aktywnymi atakami (np. wybranym szyfrogramem) ze strony innych stron w sieci.

Zgodnie z naszą dyskusją w sekcji 4.5, naturalnym pomysłem jest zastosowanie podejścia „szyfruj, a następnie uwierzytelnią”, w którym S wysyła S, c, SignsS (c) do R, gdzie c jest szyfrowaniem m przy użyciu klucza szyfrującego ekR należącego do R. (Dla wygody podajemy tutaj wyraźnie tożsamość nadawcy.) Jednakże mamy tutaj do czynienia ze sprytnym atakiem z użyciem wybranego tekstu zaszyfrowanego, niezależnie od użytego schematu szyfrowania.

Po zaobserwowaniu transmisji jak powyżej inna (kontrowersyjna) strona A może usunąć podpis S i zastąpić go własnym, wysyłając A, c, Znak A (c) do R.

W tym przypadku R nie wykryłyby niczego złego i błędnie pomyślałyby, że A wysłał mu wiadomość m. Jeśli R odpowie A lub w inny sposób zachowa się wobec A w sposób zależny od treści wiadomości, wówczas A może potencjalnie poznać nieznaną wiadomość m.

(Innym problemem związanym z tym schematem, chociaż w pewnym stopniu niezależnym od naszej dyskusji, jest to, że nie zapewnia on już niezaprzeczalności. Oznacza to, że R nie może łatwo udowodnić stronie trzeciej, że S podpisał wiadomość m, przynajmniej nie bez ujawnienia jej własny klucz deszyfrujący dkR.)

Zamiast tego można zastosować podejście „uwierzytelni, a następnie zaszyfruj”. Tutaj, s najpierw obliczy podpis σ SignsS (m), a następnie wyśle

S, EncekR (m).

(Zauważ, że rozwiązuje to wspomniany powyżej problem niezaprzeczalności.) Jeśli schemat szyfrowania jest bezpieczny tylko według CPA, wówczas występują problemy takie jak te wymienione w sekcji 4.5, więc założmy, że zamiast tego używany jest schemat szyfrowania bezpieczny CCA. Nawet wtedy istnieje atak, który może przeprowadzić złośliwy wirus R. Po otrzymaniu S, EncekR (m) od S, złośliwy R może odszyfrować w celu uzyskania m, a następnie ponownie zaszyfrować i wysłać S, EncekR (m) do innego odbiorcy R. Ten (uczciwy) odbiorca R pomyśli wówczas, że S wysłał mu wiadomość m. Może to mieć poważne konsekwencje, np. jeśli m jest komunikatem „Jestem ci winien 100 dolarów”.

Atakom tym można zapobiec, jeśli strony będą bardziej ostrożne w sposobie obchodzenia się z identyfikatorami. Podczas szyfrowania nadawca powinien dołączyć do wiadomości swoja tożsamość; podpisując, strona powinna podpisać tożsamość zamierzzonego odbiorcy wraz z tym, co jest podpisywane. Na przykład drugie podejście zostanie zmodyfikowane w taki sposób, że S najpierw oblicza σ SignsS (mR), a następnie wysyła S, EncekR (Sm) do R. Podczas deszyfrowania odbiorca powinien sprawdzić, czy wynikowa odszyfrowana wartość zawiera (rzekomo) tożsamość nadawcy; podczas weryfikacji odbiorca powinien sprawdzić, co było

podpisany zawiera własną tożsamość. Uwzględniając tożsamości w ten sposób, zarówno uwierzytelnianie, a następnie szyfrowanie, jak i szyfrowanie, a następnie uwierzytelnianie są bezpieczne, jeśli używany jest schemat szyfrowania zabezpieczony przez CCA i silnie bezpieczny schemat podpisu (którego definicja jest zgodna z definicją 4.3 dla komputerów MAC).

Referencje i dodatkowe lektury

Godne uwagi wczesne prace nad podpisami obejmują prace Diffiego i Hellmana [58], Rabina [144, 145], Rivesta, Shamira i Adlemana [148] oraz Goldwassera, Micali i Yao [82]. Obszerne omówienie schematów podpisów wykraczające poza to, co można tu omówić, można znaleźć w [98].

Goldwasser, Micali i Rivest [81] zdefiniowali pojęcie egzystencjalnej niepodrabialności w przypadku adaptacyjnego ataku wybranej wiadomości, a także przedstawili pierwszą konstrukcję stanowego schematu podpisu spełniającą tę definicję. Gold-reich [74] zasugerował podejście polegające na uczynieniu schematu Goldwassera–Micali–Rivesta bezpaństwowcem i zasadniczo przyjęliśmy pomysł Goldreicha w sekcji 12.6.3.

Zwykłe podpisy RSA pochodzą z oryginalnego papieru RSA [148]. RSA-FDH został zaproponowany przez Bellare'a i Rogawaya w ich oryginalnej pracy na temat modelu losowej wycieczki [21], chociaż pomysł (bez dowodu) wykorzystania kryptograficznej funkcji skrótu w celu zapobiegania atakom algebraicznym wywodzi się od Rabina [145]. Późniejsze ulepszenie [23] zostało ujednolicone w ramach PKCS #1 v2.1, dostępnego pod adresem <http://www.emc.com/emc-plus/rsa-labs>.

Transformata Fiata-Shamira [65] i schemat podpisu Schnorra [152] datowane są na koniec lat 80. XX wieku, chociaż nasz dowód Twierdzenia 12.10 wynika z [1], a nasz dowód Twierdzenia 12.11 jest inspirowany [20]. Standardy DSA i ECDSA opisano w [132].

Schemat podpisu Lamporta został opublikowany w 1979 r. [111], choć był już opisany w [58]. Konstrukcję opartą na drzewach, podobną w duchu do Construction 12.20, zaproponował Merkle [121, 122], chociaż podejście oparte na drzewach zastosowano także w [81]. Naor i Yung [128] pokazali, że permutacje jednokierunkowe wystarczą do skonstruowania jednorazowych podpisów bezpiecznych, które mogą podpisywać wiadomości o dowolnej długości, co zostało ulepszone przez Rompela [151], który pokazał, że funkcje jednokierunkowe są wystarczające. (Zobacz także [99].) Jak widzieliśmy w podrozdziale 12.6.3, jednorazowe bezpieczne podpisy tego rodzaju mogą być użyte do skonstruowania schematów bezpiecznych podpisów, co oznacza, że funkcje jednokierunkowe wystarczą do istnienia (bezstanowych) bezpieczne podpisy.

Pojęcie certyfikatów zostało po raz pierwszy opisane przez Kohnfeldera [107] w jego pracy licencjackiej. Infrastruktury klucza publicznego są omówione bardziej szczegółowo w [102, rozdział 15]. Zobacz także [3, 62]. Dalsze szczegóły protokołu TLS można znaleźć np. w [102, 165]. Formalne podejście do połączonej tajemnicy i integralności w kontekście klucza publicznego przedstawili An i in. [10].

Ćwiczenia

- 12.1 Pokaż, że Konstrukcja 4.7 do konstruowania MAC o zmiennej długości z dowolnego MAC o stałej długości może być również użyta (z odpowiednimi modyfikacjami) do skonstruowania schematu podpisu dla wiadomości o dowolnej długości z dowolnego schematu podpisu dla wiadomości o stałej długości (n) N .
- 12.2 Udowodnij, że istnienie schematu jednorazowego bezpiecznego podpisu dla wiadomości 1-bitowych implikuje istnienie funkcji jednokierunkowych.
- 12.3 W sekcji 12.4.1 pokazaliśmy atak na zwykły schemat podpisu RSA, w którym osoba atakująca fałszuje podpis na dowolnej wiadomości za pomocą dwóch zapytań podpisujących. Pokaż, jak osoba atakująca może sfałszować podpis na dowolnej wiadomości za pomocą pojedynczego zapytania podpisującego.
- 12.4 Założmy, że problem RSA jest trudny. Pokaż, że zwykły schemat podpisu RSA spełnia następującą słabą definicję bezpieczeństwa: atakujący otrzymuje klucz publiczny N , e i jednolity komunikat $m \in Z$. Sary odnosi sukces, jeśli może $m^e \mod N$. Przeciwnik wyprowadzić ważny podpis na m bez wykonywania żadnych zapytań o podpisanie.
- 12.5 Innym podejściem (oprócz hashowania), które zostało wypróbowane w celu skonstruowania bezpiecznych podpisów opartych na RSA, jest zakodowanie wiadomości przed zastosowaniem permutacji RSA. W tym przypadku osoba podpisująca ustala publiczną funkcję kodowania $\text{enc} : \{0, 1\}^k \rightarrow Z$ jako część swojego klucza publicznego, a podpis w $\text{Wiadomości } m$ to $\sigma := [\text{enc}(m)]^k$
- (a) W jaki sposób przeprowadzana jest weryfikacja w zakodowanym formacie RSA? (b) Omów, dlaczego odpowiedni wybór funkcji kodowania dla N zapobiega „atakowi bez wiadomości” opisanemu w Sekcji 12.4.1. (c) Pokaż, że zakodowany RSA jest niepewny, jeśli $\text{enc}(m) = 0x00m0 \dots 0x00k/10 = 4k$ (gdzie $k \stackrel{\text{def}}{=} N, = |m|$) i m nie jest komunikatem składającym się wyłącznie z 0). Założmy, że $e = 3$.
- (d) Pokaż, że zakodowany RSA jest niepewny dla $\text{enc}(m) = 0m0m \dots 0m0m$ (gdzie $= (N - 1)/2$ i $m = |m|$ nie jest komunikatem składającym się wyłącznie z 0). Przypuszczać $mi = 3$.
- (e) Rozwiąż części (c) i (d) dla dowolnego e .
- 12.6 Rozważmy wariant transformacji Fiata-Shamira, w którym podpisem jest (I, s) , a nie (r, s) , a weryfikacja zmienia się w sposób naturalny. Pokaż, że jeśli podstawowy schemat identyfikacji jest bezpieczny, to wynikowy schemat podpisu jest również tutaj bezpieczny.

12.7 Rozważmy wariant DSA, w którym przestrzeń komunikatów to Z_q , a H jest pominięte. (Tak więc drugim składnikiem podpisu jest teraz $s := [k$

$^1 \cdot (m + xr) \bmod q]$.) Pokaż, że ten wariant nie jest bezpieczny.

12.8 Niech f będzie permutacją jednokierunkową. Rozważ następujący podpis schemat wiadomości w zestawie $\{1, \dots, N\}$:

- Aby wygenerować klucze, wybierz uniform $x \in \{0, 1\}^n$ i ustaw $y := f(n)(x) = x$. The def (gdzie $f(i)(\cdot)$ odnosi się do i -krotnej iteracji f , a $f(0)(x)$ kluczem publicznym jest y , a kluczem prywatnym jest x .

Aby podpisać wiadomość $i \in \{1, \dots, n\}$, wyjście $f(n-i)(x)$ • Aby zweryfikować podpis σ na wiadomości i i w odniesieniu do klucza publicznego y , sprawdź, czy $y \stackrel{?}{=} f(i)(\sigma)$.

(a) Wykaż, że powyższy schemat nie jest schematem jednorazowego bezpiecznego podpisu.

Biorąc pod uwagę podpis w wiadomości i , dla jakich wiadomości j przeciwnik może sfałszować? (b) Udowodnić, że żaden

przeciwnik ppt , któremu podpisano podpis na i , nie może sfałszować żadnej wiadomości $j > i$, chyba że prawdopodobieństwo jest znikome. (c) Zaproponuj, jak zmodyfikować schemat, aby uzyskać schemat podpisu jednorazowego bezpiecznego.

Wskazówka: Uwzględnij dwie wartości y , y w kluczu publicznym.

12.9 Schemat silnego jednorazowego bezpiecznego podpisu spełnia (nieformalnie): mając podpis σ w wiadomości m , nie jest możliwe wyrowadzenie $(m, \sigma) = (m, \sigma)$, dla którego σ jest prawidłowym podpisem na m (zauważ, że $m = m$ jest dozwolone).

(a) Podaj formalną definicję silnych, jednorazowych podpisów. (b) Zakładając istnienie funkcji jednokierunkowych, pokaż funkcję jednokierunkową, dla której schemat Lamporta nie jest silnym schematem podpisu jednorazowego. (c) Skonstruuj silny schemat jednorazowego

bezpiecznego podpisu w oparciu o założenia zastosowane w tej książce.

Wskazówka: użyj określonej funkcji jednokierunkowej w schemacie Lamporta.

12.10 Rozważmy schemat podpisu Lamporta. Opisz przeciwnika, który uzyskuje podpisy pod dwiema wybranymi przez siebie wiadomościami, a następnie może sfałszować podpisy pod dowolną wiadomością.

12.11 Schemat Lamporta wykorzystuje 2 wartości klucza publicznego do podpisywania wiadomości o długości r . Rozważmy wariant, w którym klucz prywatny zawiera 2 wartości x_1, \dots, x_2 , a klucz publiczny zawiera wartości y_1, \dots, y_2 z $y_i := f(x_i)$. Wiadomość $m \in \{0, 1\}^r$ jest odwzorowywana w sposób jeden do jednego do podzbioru $S_m \subseteq \{1, \dots, 2\}^r$ o rozmiarze r . Aby podpisać m , podpisujący odkrywa $\{x_i\} \subseteq S_m$. Udowodnij, że daje to schemat jednorazowego bezpiecznego podpisu.

Jaka jest maksymalna długość wiadomości obsługiwana przez ten schemat?

12.12 Na końcu sekcji 12.6.3 pokazujemy, jak można użyć funkcji pseudolosowej, aby konstrukcja 12.20 stała się bezstanowa. Czy podobne podejście sprawdza się w przypadku schematu opartego na łańcuchu opisanego w sekcji 12.6.2? Jeśli tak, naszkicuj konstrukcję i dowód. Jeśli nie, wyjaśnij dlaczego i zmodyfikuj schemat, aby uzyskać wariant bezstanowy.

12.13 Dowód twierdzenia 12.22.

12.14 Założymy, że unieważnianie certyfikatów odbywa się w następujący sposób: gdy użytkownik Bob twierdzi, że klucz prywatny odpowiadający jego kluczowi publicznemu pkB został skradziony, użytkownik przesyła do urzędu certyfikacji oświadczenie o tym fakcie podpisane w odniesieniu do pkB. Po otrzymaniu tak podpisanej wiadomości urząd certyfikacji unieważnia odpowiedni certyfikat.

Wyjaśnij, dlaczego w tym przypadku organ właściwy nie musi sprawdzać tożsamości Boba. W szczególności wyjaśnij, dlaczego nie przejmuje się tym, że przeciwnik, który ukradł klucz prywatny Boba, może sfałszować podpisy w odniesieniu do pkB.

Rozdział 13

*Zaawansowane tematy w kluczu publicznym Szyfrowanie

W rozdziale 11 widzieliśmy kilka przykładów schematów szyfrowania kluczem publicznym stosowanych w praktyce. W tym miejscu zbadamy niektóre schematy, które obecnie cieszą się większym zainteresowaniem teoretycznym, choć w niektórych przypadkach możliwe jest, że te schematy (lub ich warianty) będą w przyszłości szerzej stosowane.

Zaczynamy od omówienia permutacji z zapadnią, uogólnienia permutacji jednokierunkowych i pokażemy, jak ich używać do konstruowania schematów szyfrowania z kluczem publicznym. Permutacje zapadni starannie odwzorowują kluczowe cechy permutacji RSA, które czynią ją tak użyteczną. Jako takie często stanowią użyteczną abstrakcję przy projektowaniu nowych kryptosystemów.

Następnie przedstawiamy trzy schematy oparte na problematyce związanej z faktoringiem:

- Schemat szyfrowania Pailliera jest przykładem schematu szyfrowania, który jest homomorficzny. Właściwość ta okazuje się przydatna do konstruowania bardziej złożonych protokołów kryptograficznych, o czym pokrótko porozmawiamy w podrozdziale 13.3.
- Schemat szyfrowania Goldwasser-Micali ma znaczenie historyczne jako pierwszy schemat, którego bezpieczeństwo CPA zostało udowodnione. Jest również homomorficzny i wykorzystuje interesującą teorię liczb, którą można zastosować w innych kontekstach.
- Na koniec omówimy permutację zapadni Rabina, która można wykorzystać do skonstruowania schematu szyfrowania z kluczem publicznym. Choć z pozoru podobna do permutacji zapadni RSA, permutacja zapadni Rabina wyróżnia się tym, że jej bezpieczeństwo opiera się bezpośrednio na twardości faktoringu. (Przypomnijmy sobie z sekcji 8.2.5, że trudność problemu RSA wydaje się silniejszym założeniem.)

13.1 Szyfrowanie z permutacji zapadni

W sekcji 11.5.3 widzieliśmy, jak skonstruować schemat szyfrowania klucza publicznego zabezpieczonego CPA w oparciu o założenie RSA. Destylując te właściwości

RSA użyte w konstrukcji i definiując abstrakcyjne pojęcie zawierające te właściwości, otrzymujemy ogólny szablon do konstruowania bezpiecznych schematów szyfrowania w oparciu o dowolny element pierwotny spełniający ten sam zestaw właściwości. Permutacje zapadni okazują się tutaj „właściwą” abstrakcją.

W następnej sekcji definiujemy (rodziny) permutacji zapadni i zauważamy, że rodzina permutacji jednokierunkowych RSA (Konstrukcja 8.77) spełnia dodatkowe wymagania, aby była rodziną permutacji zapadni. W podrozdziale 13.1.2 uogólniamy konstrukcję z podrozdziału 11.5.3 i pokazujemy, że szyfrowanie kluczem publicznym można skonstruować na podstawie dowolnej permutacji zapadni. Wyniki te zostaną ponownie wykorzystane w rozdziale 13.5, gdzie pokazujemy drugi przykład permutacji zapadni, tym razem oparty bezpośrednio na założeniach faktoringu.

W tej sekcji opieramy się na materiale z rozdziału 8.4.1 lub alternatywnie z rozdziału 7.

13.1.1 Permutacje zapadni

Przypomnijmy definicje rodzin funkcji i rodzin permutacji jednokierunkowych z podrozdziału 8.4.1. W tej sekcji pokazaliśmy, że założenie RSA w naturalny sposób prowadzi do rodziny jednokierunkowych permutacji. Wnikliwy czytelnik mógł zauważyc, że podana przez nas konstrukcja (Konstrukcja 8.77) ma specjalną właściwość, o której tam nie wspomniano: mianowicie algorytm generowania parametrów Gen generuje dodatkowe informacje wraz z I, które umożliwiają wydajną inwersję f_I . Takie dodatkowe informacje nazywamy zapadnią i za pomocą tej dodatkowej właściwości nazywamy rodzinę permutacji jednokierunkowych rodzinami permutacji zapadni. Poniżej znajduje się formalna definicja.

DEFINICJA 13.1 Krotka algorytmów czasu wielomianowego (Gen , Samp , f , Inv) jest rodziną permutacji zapadni (lub permutacji zapadni), jeśli:

- Algorytm probabilistycznej generacji parametrów Gen , na wejściu 1^n , wyjściach (I, td) z $|I| = r$. Każda wartość I definiuje zbiór D_I , który stanowi dziedzinę i zakres permutacji (tj. bijekcji) $f_I : D_I \rightarrow D_I$.
- Niech Gen_1 oznacza algorytm powstały w wyniku uruchomienia Gen i wygenerowania tylko I . Wtedy $(\text{Gen}_1, \text{Samp}, f)$ jest rodziną jednokierunkowych permutacji.
- Niech (I, td) będzie wyjściem $\text{Gen}(1^n)$. Deterministyczny algorytm odwracający Inv , na wejściu td i $y \in D_I$, daje na wyjściu $x \in D_I$. Oznaczamy to przez $x := \text{Inv}_{td}(y)$. Wymagane jest, aby z prawie znikomym prawdopodobieństwem ponad (I, td) wyjście przez $\text{Gen}(1^n)$ i równomierny wybór $x \in D_I$, mamy

$$\text{Inv}_{td}(f_I(x)) = x.$$

W skrócie pomijamy wyraźną wzmiankę o Samp i po prostu odnosimy się do permutacji zapadni (Gen , f , Inv). Dla wyjścia (I, td) Gen zapisujemy $x \in D_I$ do

oznaczają równomierny wybór $x \in DI$ (przy założeniu, że robi to algorytm Samp).

Drugi warunek powyżej implikuje, że fI nie może zostać skutecznie odwrócony bez td , ale ostatni warunek oznacza, że fI może zostać skutecznie odwrócony za pomocą td . Jest oczywiste, że Konstrukcję 8.77 można zmodyfikować, aby uzyskać rodzinę permutacji zapadni, jeśli problem RSA jest trudny w porównaniu z GenRSA, dlatego nazywamy tę konstrukcję permutacją zapadni RSA.

13.1.2 Szyfrowanie kluczem publicznym na podstawie permutacji zapadni

Teraz naszkicujemy, jak można zbudować schemat szyfrowania z kluczem publicznym na podstawie dowolnej rodziny permutacji zapadni. Konstrukcja jest po prostu uogólnieniem tego, co zostało już zrobione dla konkretnej permutacji zapadni RSA w sekcji 11.5.3.

Zaczniemy od (ponownego) wprowadzenia pojęcia twardego predykatu. Jest to naturalna adaptacja Definicji 7.4 do naszego kontekstu, a także uogólnienie naszej wcześniejszej dyskusji na temat jednego konkretnego, twardego predykatu dla permutacji zapadni RSA w Podrozdziale 11.5.3.

DEFINICJA 13.2 Niech $\Pi = (\text{Gen}, f, \text{Inv})$ będzie rodziną per-mutacji zapadni i niech hc będzie deterministycznym algorytmem wielomianowym w czasie, który na wejściu I i $x \in DI$ daje na wyjściu pojedynczy bit $hcI(x)$. Mówimy, że hc jest twardym predykatem Π , jeśli dla każdego probabilistycznego algorytmu A w czasie wielomianowym istnieje pomijalna funkcja negl taka, że

$$\Pr[A(I, fI(x)) = hcI(x)] = \frac{1}{2} + \text{negligible}(n),$$

gdzie prawdopodobieństwo przejmuje się eksperymentem, w którym $\text{Gen}(1n)$ jest uruchamiany w celu wygenerowania (I, td) , a następnie x jest wybierane równomiernie z DI .

Asimetria zapewniana przez permutacje zapadni oznacza, że każdy, kto zna zapadnię td powiązaną z I , może odzyskać x z $fI(x)$, a tym samym obliczyć $hcI(x)$ z $fI(x)$. Ale mając tylko I , nie jest możliwe obliczenie $hcI(x)$ z $fI(x)$ dla jednolitego x .

Poprzez odpowiednią modyfikację Twierdzenia 7.5 można udowodnić, co następuje:

TWIERDZENIE 13.3. Mając daną rodzinę permutacji zapadni Π , istnieje rodzina permutacji zapadni Π z twardym predykatem hc dla Π .

Mając rodzinę permutacji zapadni $\Pi = (\text{Gen}, f, \text{Inv})$ z twardym predykatem hc , możemy skonstruować jednobitowy schemat szyfrowania za pomocą następującego podejścia (patrz Konstrukcja 13.4 poniżej i porównaj z Konstrukcją 11.32): klucze, uruchom $\text{Gen}(1n)$, aby uzyskać (I, td) ; klucz publiczny to ja i the

klucz prywatny to td . Mając klucz publiczny I , szyfrowanie wiadomości $m \in \{0, 1\}$ polega na wybraniu jednolitego $r \in DI$ pod warunkiem, że $hcI(r) = m$, a następnie ustaleniu tekstu zaszyfrowanego równego $fI(r)$. Aby odszyfrować, odbiorca używa td do odzyskania r z $fI(r)$, a następnie wysyła wiadomość $m := hcI(r)$.

BUDOWA 13.4

Niech $\Pi = (\text{Gen}, f, \text{Inv})$ będzie rodziną permutacji zapadni z twardym predykatorem hc . Zdefiniuj schemat szyfrowania klucza publicznego w następujący sposób:

- **Gen:** na wejściu $1n$ klucz, uruchom $\text{Gen}(1n)$, aby uzyskać (I, td) . Wyrowadź opinię publiczną I i klucz prywatny td .
- **Enc:** po wprowadzeniu klucza publicznego I i wiadomości $m \in \{0, 1\}$, wybierz jednolite $r \in DI$ z zastrzeżeniem ograniczenia, że $hcI(r) = m$. Wyrowadź zaszyfrowany tekst $c := fI(r)$.
- **Dec:** po wprowadzeniu klucza prywatnego td i tekstu zaszyfrowanego c , oblicz wartość $r := \text{Inv}(c)$ i wyślij wiadomość $hcI(r)$.

Szyfrowanie kluczem publicznym z dowolnej rodziny permutacji zapadni.

Dowód bezpieczeństwa przebiega analogicznie do dowodu Twierdzenia 11.33.

TWIERDZENIE 13.5 Jeśli Π jest rodziną permutacji zapadni z twardym predykatorem hc , to Konstrukcja 13.4 jest zabezpieczona CPA.

DOWÓD Niech Π oznacza konstrukcję 13.4. Udowodnimy, że Π ma nieroróżnialne szyfrowanie w obecności podsłuchującego; według Propozycji 11.3 oznacza to, że jest on bezpieczny dla CPA.

Najpierw zauważamy, że hc musi być bezstronny w następującym sensie. Pozwalać

$$\delta_0(n) \stackrel{\text{def}}{=} \Pr_{(I, \text{td})}[\text{Gen}(1n); x \in DI] [hcI(x) = 0]$$

$$\delta_1(n) \stackrel{\text{def}}{=} \Pr_{(I, \text{td})}[\text{Gen}(1n); x \in DI] [hcI(x) = 1].$$

Wtedy istnieje pomijalna funkcja negl taka, że

$$1 \leq \delta_0(n), \delta_1(n) \leq \text{negl}(n);$$

jeśli nie, wówczas atakujący, który po prostu wprowadza częściej występujący bit, naruszyłby definicję 13.2.

Niech teraz A będzie probabilistycznym przeciwnikiem wielomianowym w czasie. Bez utraty ogólności w eksperymencie PubKeav $A, \Pi(n)$ możemy przyjąć $m_0 = 0$ i $m_1 = 1$.

Następnie mamy

$$\Pr[\text{PubKeav } A, \Pi(n) = 1] = \frac{1}{2} \cdot \Pr[A(\text{pk}, c) = 0 \mid c \text{ jest szyfrowaniem 0}] \\ + \frac{1}{2} \cdot \Pr[A(\text{pk}, c) = 1 \mid c \text{ jest szyfrowaniem 1}].$$

Ale wtedy

$$\Pr[A(I, f^I(x)) = h^I(x)] = \\ 80(n) \cdot \Pr[A(I, f^I(x)) = 0 \mid h^I(x) = 0] \\ + \delta 1(n) \cdot \Pr[A(I, f^I(x)) = 1 \mid h^I(x) = 1] \\ - \frac{1}{2} \cdot \text{negl}(n) \cdot \Pr[A(I, f^I(x)) = 0 \mid h^I(x) = 0] \\ + \frac{1}{2} \cdot \text{negl}(n) \cdot \Pr[A(I, f^I(x)) = 1 \mid h^I(1) = 1] \\ - \frac{1}{2} \cdot \Pr[A(I, f^I(x)) = 0 \mid h^I(x) = 0] 2 \\ - \frac{1}{2} \cdot \Pr[A(I, f^I(x)) = 1 \mid h^I(1) = 1] - 2 \cdot \text{negl}(n) + 2 \\ = \Pr[\text{PubKeav } A, \Pi(n) = 1] - 2 \cdot \text{negl}(n).$$

Ponieważ h^I jest twardym predykatem dla Π , istnieje pomijalna funkcja negl taka, że $\text{negl}(n) = \Pr[A(I, f^I(x)) = h^I(x)]$; to znaczy że

$$\Pr[\text{PubKeav } A, \Pi(n) = 1] = \text{negl}(n) + 2 \cdot \text{negl}(n),$$

uzupełnienie dowodu. ■

Szyfrowanie dłuższych wiadomości. Korzystając z Twierdzenia 11.7, wiemy, że możemy rozszerzyć Konstrukcję 13.4, aby szyfrować wiadomości bitowe przy użyciu tekstów zaszyfrowanych razy dłuższych. Większą wydajność można uzyskać, konstruując KEM, postępując zgodnie z wytycznymi konstrukcji 11.34. Szczegóły pozostawiamy jako ćwiczenie.

13.2 Schemat szyfrowania Paillier

W tej sekcji opisujemy schemat szyfrowania Pailliera, schemat szyfrowania z kluczem publicznym, którego bezpieczeństwo opiera się na założeniach związanych (ale nie wiadomo, czy są równoważne) z trudnością faktoringu. Ten schemat szyfrowania jest szczególnie interesujący, ponieważ ma kilka dobrych właściwości homomorficznych, co omówimy dalej w sekcji 13.2.3.

Schemat szyfrowania Paillier wykorzystuje grupę Z_{N^2} , multiplikatywną grupę elementów z zakresu $\{1, \dots, N^2\}$, które są względnie pierwsze względem N , dla N jest iloczynem dwóch różnych liczb pierwszych. Aby zrozumieć schemat, pomocne jest najpierw zrozumienie struktury Z_{N^2} . Przydatna charakterystyka tej grupy daje następujące twierdzenie, które mówi między innymi (por. Definicja 8.23) dla N o postaci, która będzie nas w Z_{N^2} jest izomorficzny z $Z_N \times Z_N$ interesować Z_{N^2} . Twierdzenie to udowodnimy następnym podrozdziałem. (Czytelnik chcący zaakceptować propozycję dotyczącej wiary może przejść do sekcji 13.2.2.)

TWIERDZENIE 13.6 Niech $N = pq$, gdzie p, q są różnymi liczbami pierwszymi nieparzystymi o równej długości. Następnie:

1. $\gcd(N, \varphi(N)) = 1$.

2. Dla dowolnej liczby całkowitej $a \neq 0$ mamy $(1 + aN) \equiv (1 + aN) \pmod{N^2}$.

a W konsekwencji rząd $(1+N)$ w Z_{N^2} to N . Oznacza to, że $(1+N) \in \langle 1 \pmod{N^2} \rangle$ i $(1+N)^N \equiv 1 \pmod{N^2}$ dla dowolnego $1 \leq a < N$.

3. $Z_{N^2} \cong Z_N \times Z_N$ jest izomorficzny z Z_{N^2} , z izomorfizmem $f : Z_{N^2} \rightarrow Z_N \times Z_N$ podane przez $f(a, b) = [(1 + aN) a \cdot b \pmod{N^2}]$.

W świetle ostatniej części powyższego twierdzenia wprowadzamy wygodną notację. Przy zrozumieniu N i $x \in Z_{N^2}$ napisz $x \in (a, b)$ jeśli $f(a, b) = x$ gdzie f jest izomorfizmem z powyższego twierdzenia. Można myśleć o tym zapisie w ten sposób, że oznacza on „ x w Z_{N^2} odpowiada (a, b) w książce $Z_N \times Z_N$ odniesieniu do izomorfizmu $Z_{N^2} \rightarrow Z_N \times Z_N$ podanego przez chińskie twierdzenie N ”. Użyliśmy w tym przypadku tego samego zapisu o resztach; zachowujemy zapis, gdyż w obu przypadkach odnosi się on do izomorfizmu grup. Niemniej jednak nie powinno być żadnych nieporozumień, ponieważ grupa Z_{N^2} i powyższa propozycja są używane tylko w tej sekcji. Zauważamy, że N^2 tutaj izomorfizm – ale nie jego odwrotność – można efektywnie obliczyć nawet bez faktoryzacji N .

13.2.1 Struktura Z_{N^2}

Ta część poświęcona jest dowodowi Twierdzenia 13.6. Przez cały czas pozwalamy N, p, q będą takie jak w twierdzeniu.

Twierdzenie 13.7 $\gcd(N, \varphi(N)) = 1$.

DOWÓD Przypomnijmy, że $\varphi(N) = (p - 1)(q - 1)$. Założymy, że $p > q$ bez utraty ogólności.

Ponieważ p jest liczbą pierwszą i $p > p - 1 > q - 1$, wyraźnie widać, że $\gcd(p, \varphi(N)) = 1$.

Podobnie $\gcd(q, q - 1) = 1$. Teraz, jeśli $\gcd(q, p - 1) = 1$ to $\gcd(q, p - 1) = q$

ponieważ q jest liczbą pierwszą. Ale wtedy $(p - 1)/q = 2$, co jest sprzeczne z założeniem, że p i q mają tę samą długość. \blacksquare

Twierdzenie 13.8 Dla liczby całkowitej $a \neq 0$ mamy $(1 + N) a \equiv 1 + N \pmod{N^2}$.
rząd $(1 + N)$ w \mathbb{Z}_N jest N .

DOWÓD Korzystając z twierdzenia o rozwinięciu dwumianowym (Twierdzenie A.1):

$$(1 + N) a = \sum_{j=0}^{A-1} \binom{A}{j} N^j.$$

Redukując prawy modulo N^2 , wszystkie wyrazy z $j > 2$ stają się 0 i $= 1 + aN \pmod{N^2}$.
zatem $a = N$. Najmniejsze niezerowe a takie, że $(1 + N) a \equiv 1 \pmod{N^2}$ wynosi $=$ \blacksquare

Twierdzenie 13.9 Izomorfizm grupy $\mathbb{Z}_N \times \mathbb{Z}_N$ jest izomorficzny z grupą \mathbb{Z}_{N^2} ,
 $f : \mathbb{Z}_N \times \mathbb{Z}_N \rightarrow \mathbb{Z}_{N^2}$ dane przez $f(a, b) = [(1 + N) a \cdot b] \pmod{N^2}$.

DOWÓD Zauważ, że $(1 + N) a \cdot b$ nie ma wspólnego dzielnika z N^2 ponieważ $\gcd((1 + N), N^2) = 1$ i $\gcd(b, N^2) = 1$ (ponieważ $b \in \mathbb{Z}_N$).
Zatem N^2 ma $\varphi(N^2)$ jedynkami. Ustalimy teraz, że f jest izomorfizmem.

Najpierw pokażemy, że f jest bijekcją. Od

$$\begin{aligned} |\mathbb{Z}_{N^2}| &= \varphi(N^2) = p \cdot (p - 1) \cdot q \cdot (q - 1) = pq \cdot (p - 1)(q - 1) \\ &= |\mathbb{Z}_N| \cdot |\mathbb{Z}_N| = |\mathbb{Z}_N \times \mathbb{Z}_N| \end{aligned}$$

(patrz Twierdzenie 8.19 dla drugiej równości), wystarczy pokazać, że f jest jeden do jednego. Powiedzmy, że $a_1, a_2 \in \mathbb{Z}_N$ i $b_1, b_2 \in \mathbb{Z}_N$ są takie, że $f(a_1, b_1) = f(a_2, b_2)$.

Następnie:

$$(1 + N) a_1 \cdot a_2 \cdot (b_1/b_2) \equiv 1 \pmod{N^2}. \quad (13.1)$$

(Zauważ, że $b_2 \in \mathbb{Z}_N$ i stąd $b_2 \in \mathbb{Z}_{N^2}$, i tak b_2 ma multiplikatywną odwrotność modulo N^2 .) Podnosząc obie strony do potęgi $\varphi(N)$ i korzystając z faktu, że rzęd Z_{N^2} wynosi $\varphi(N^2) = N \cdot \varphi(N)$ otrzymujemy

$$\begin{aligned} (1 + N) (a_1 - a_2) \cdot \varphi(N) \cdot (b_1/b_2)^{\varphi(N)} &\equiv 2 \pmod{N^2} \\ (1 + N) (a_1 - a_2) \cdot \varphi(N) &\equiv 2 \pmod{N^2}. \end{aligned}$$

Zgodnie z zastrzeżeniem 13.8, $(1 + N)$ ma rzęd N modulo N^2 . Stosując Twierdzenie 8.53, widzimy, że $(a_1 - a_2) \cdot \varphi(N) \equiv 0 \pmod{N}$, a więc N dzieli $(a_1 - a_2) \cdot \varphi(N)$.

Ponieważ $\gcd(N, \varphi(N)) = 1$ według zastrzeżenia 13.7, wynika z tego, że $N \mid (a_1 - a_2)$. Ponieważ $a_1, a_2 \in \mathbb{Z}_N$, może to nastąpić tylko wtedy, gdy $a_1 = a_2$.

Wracając do równania (13.1) i ustawiając $a_1 = a_2$, mamy zatem $b \bmod N$. Ponieważ $b_1^N \equiv b_2^N \pmod{N^2}$. Oznacza to, że $b_1^N = b_2^N \pmod{N}$ jest względnie pierwszą liczbą $\varphi(N)$, rząd Z_N (por. Wniosek 8.17). Oznacza to, że $b_1 = b_2 \pmod{N}$; ponieważ $b_1, b_2 \in Z$ mają $b_1 = b_2$. Dochodzimy, my do wniosku, że f jest różnowartością, a zatem jest bijekcją.

Aby pokazać, że f jest izomorfizmem, pokazujemy, że $f(a_1, b_1) \cdot f(a_2, b_2) = f(a_1 + a_2, b_1 + b_2)$. (Zauważ, że mnożenie po lewej stronie równości odbywa się modulo N^2 , podczas gdy dodawanie/mnożenie po prawej stronie odbywa się modulo N). Mamy:

$$\begin{aligned} f(a_1, b_1) \cdot f(a_2, b_2) &= (1 + N) a_1 \cdot b_1^{N-1} \cdot (1 + N) a_2 \cdot b_2^{N-1} \pmod{N^2} \\ &= (1 + N)^{a_1+a_2} \cdot (b_1 b_2)^{N-2} \pmod{N^2}. \end{aligned}$$

Ponieważ $(1 + N)$ ma rząd N modulo N^2 (zgodnie z Twierdzeniem 13.8), możemy zastosować Propozycja 8.52 i uzyskać

$$\begin{aligned} f(a_1, b_1) \cdot f(a_2, b_2) &= (1 + N)^{a_1+a_2} \cdot (b_1 b_2)^{N-2} \pmod{N^2} \\ &= (1 + N) [a_1 + a_2 \pmod{N}] \cdot (b_1 b_2)^{N-2} \pmod{N^2}. \end{aligned} \quad (13.2)$$

Jeszcze nie skończyliśmy, ponieważ $b_1 b_2$ w równaniu (13.2) reprezentuje mnożenie modulo N^2 , podczas gdy chcielibyśmy, żeby było to modulo N . Niech $b_1 b_2 = r + yN$, gdzie y, r są liczbami całkowitymi, gdzie $0 \leq r < N$ (r nie może być 0 ponieważ $b_1, b_2 \in Z$, $N \geq 1$ więc ich iloczyn nie jest podzielny przez N). Zauważ, że $r = b_1 b_2 \pmod{N}$. Mamy również

$$\begin{aligned} (b_1 b_2)^N &= (r + yN)^N \pmod{N^2} \\ &= \sum_{k=0}^{N-1} \binom{N}{k} r^N k^k (yN)^k \pmod{N^2} \\ &= r^N + N \cdot r^{N-1} \cdot (yN) = r^N = ([b_1 b_2 \pmod{N}]N)N \pmod{N^2}, \end{aligned}$$

stosując twierdzenie o rozszerzaniu dwumianowym jak w zastrzeżeniu 13.8. Podstawiając to do równania (13.2) otrzymujemy pożądany wynik:

$$\begin{aligned} f(a_1, b_1) \cdot f(a_2, b_2) &= (1 + N) [a_1 + a_2 \pmod{N}] \cdot (b_1 b_2 \pmod{N}) \pmod{N^2} \\ &= f(a_1 + a_2, b_1 + b_2), \end{aligned}$$

udowadniając, że f jest izomorfizmem $Z_N \times Z \rightarrow Z_N$. ■

13.2.2 Schemat szyfrowania Paillier

Niech $N = pq$ będzie iloczynem dwóch różnych liczb pierwszych o równej długości. Twierdzenie 13.6 mówi, że Z_N jest izomorficzny z Z_{N^2} , z izomorfizmem danym modulo Z przez $f(a, b) = [(1 + N) a \cdot b] \pmod{N^2}$. Konsekwencją jest to, że jest to element jednolity

y $\in Z_{N^2}$ odpowiada elementowi jednorodnemu $(a, b) \in ZN \times Z^N$ lub w innym słów, element (a, b) o uniformie $a \in ZN$ i uniformie $b \in Z^N$ an N-ta reszta $\in N$.

Oznaczamy N^2 modulo N^2 jeśli y jest N-tą potegą, czyli zadzwoń y $\in Z$ mod N^2 . Scharakteryzuje $[r \in Z]$ dla $r \in Z$ istnieje $y \in Z$ z $y \equiv r^N \pmod{N^2}$ tzn. $y = r^N \pmod{N^2}$ dla dowolne $x \in Z$

N^2 gdzie $x \in (a, b)$ i podniesienie go do N-tej potęgi daje:

$$[x^{N^2 \bmod N}] \cdot (a, b)^N = (N \cdot a \bmod N, bN \bmod N) = (0, bN \bmod N).$$

(Przypomnijmy, że operacja grupowa na składowej $ZN \times Z^N$ jest dodawaniem modulo N w pierwszym $ZN \times Z^N$ i mnożenie modulo N w drugiej składowej.) Ponadto twierdzimy, że dowolny element y z y $\equiv (0, b)$ jest N-tą resztą. Aby to zobaczyć, przypomnij sobie, że $\gcd(N, \varphi(N)) = 1$ i tak d

$\stackrel{\text{def}}{=} [r \in Z] \quad r^{-1} \bmod \varphi(N)$ istnieje. Więc

$$(a, [b^D \bmod N])N = (N \cdot a \bmod N, [ur^{-D} \bmod N]) = (0, b) \quad y$$

dla dowolnego $a \in ZN$. Pokazaliśmy zatem, że $\text{Res}(N^2)$ odpowiada zbiorowi

$$(0, b) | b \in Z_N.$$

Powyższe pokazuje również, że liczba N-tych pierwiastków dowolnego y $\in \text{Res}(N^2)$ wynosi dokładnie N, więc obliczanie N-tych potęg jest funkcją N do 1. Zatem, jeśli $r \in Z$ jest jednorodny, to $[r \in Z]$ grubsza $\stackrel{N}{\bmod} N^2$ jest jednolitym elementem $\text{Res}(N^2)$.

mówiąc, decyzyjny problem złożonej rezystwności polega na odróżnieniu od jednorodny element Z, niech $\text{Res}(N^2)$ jednolitego elementu $\text{Res}(N^2)$. Formalnie, utwórz GenModulus będzie algorytmem działającym w czasie wielomianowym, który na wejściu $1n$ daje (N, p, q) , gdzie $N = pq$ oraz p i q są n-bitowymi liczbami pierwszymi (z wyjątkiem prawdopodobieństwa zaniedbywalnego w N). Następnie:

DEFINICJA 13.10 Decyzyjny problem resztości złożonej jest trudny w porównaniu z GenModulus, jeśli dla wszystkich probabilistycznych algorytmów wielomianowych D istnieje funkcja pomijalna taka, że

$$\Pr[D(N, [r \in Z_N]) = 1] = \Pr[D(N, r) = 1] \quad \text{negl}(n),$$

gdzie w każdym przypadku prawdopodobieństwa przejmuje się z eksperymentu, w którym GenModulus($1n$) wyprowadza (N, p, q) , a następnie wybiera się jednorodność $r \in Z$ mod N^2 (Przypomnijmy, że $[r \in Z]$ jest jednolitym elementem $\text{Res}(N^2)$.)

Założenie decyzyjnej złożonej rezystwności (DCR) to założenie, że istnieje GenModulus, względem którego decyzyjny złożony problem rezydualności jest trudny.

Jak już omawialiśmy, elementy Z mają postać $\stackrel{N^2}{(r, r)}$, gdzie $r \in Z$ są dowolne (w odpowiednich grupach), natomiast N-te reszty mają postać $(0, r)$, gdzie $r \in Z$ są dowolne. Założenie DCR jest takie, że jest to trudne

odróżniać elementy jednolite pierwszego typu od jednolitych elementów drugiego typu. Sugeruje to następujący abstrakcyjny sposób szyfrowania wiadomości $m \in Z_N$ w odniesieniu do klucza publicznego N : wybierz jednolitą N -tą resztę $(0, r)$ i ustaw szyfrogram równy

$$\text{do } (m, 1) \cdot (0, r) = (m + 0, 1 \cdot r) = (m, r).$$

Nie martwiąc się na razie, jak skutecznie może to zrobić nadawca lub jak odszyfrować odbiorca, po prostu przekonajmy się (na poziomie intuicyjnym), że jest to bezpieczne. Ponieważ jednolitej N -tej reszty $(0, r)$ nie można odróżnić od jednolitego elementu (r, r) , szyfrogram skonstruowany powyżej jest nie do odróżnienia (z punktu widzenia podsłuchującego, który nie zna faktoryzacji N) od tekstu zaszyfrowanego

$$\text{do } (m, 1) \cdot (r, r) = ([m + r \bmod N], r)$$

dla jednorodnych $r \in Z_N$ i $r \in Z_N$. Lemat 11.15 pokazuje, że $[m + r \bmod N]$ jest równomiernie rozłożone w Z_N , a więc w szczególności ten zaszyfrowany tekst c jest niezależny od wiadomości m . Następuje bezpieczeństwo CPA. Formalny dowód, który przebiega dokładnie według tych zasad, jest podany poniżej.

Zanim przejdziemy do opisu formalnego i dowodu bezpieczeństwa, pokażemy, jak to zrobić szyfrowanie i deszyfrowanie może być wykonywane skutecznie.

Szyfrowanie. Szyfrowanie opisaliśmy powyżej tak, jakby odbywało się ono w $Z_N \times Z$, nadawca generuje szyfrogram \mathbf{N} . Faktycznie ma to miejsce w grupie izomorficznej Z_{N^2} . Że wybierając $c \in Z_{N^2}$, a następnie oblicza

$$\text{mundur1 } r \in Z_N$$

$$\text{do} := [(1 + N)m \in Z_N \bmod N]$$

Obseruj to

$$\text{do} = (1 + N)m \cdot 1 \in Z_N \cdot (1 + N)^0 \cdot r \in Z_N \bmod N = (m, 1) \cdot (0, r),$$

i tak $c \in (m, r)$ zgodnie z potrzebą.

Odszyfrowanie. Opiszemy teraz, jak można efektywnie przeprowadzić deszyfrowanie, biorąc pod uwagę faktoryzację N . Dla c skonstruowanego jak powyżej twierdzimy, że m jest odzyskiwane w następujących krokach:

- Ustaw $\hat{c} := [c \varphi(N) \bmod N^2]$.
- Ustaw $\hat{m} := (\hat{c} - 1)/N$. (Zauważ, że odbywa się to na liczbach całkowitych.)
- Ustaw $m := \hat{m} \cdot \varphi(N)^{-1} \bmod N$.

¹Zauważamy, że nie ma znaczenia, czy nadawca wybierze jednolity $r \in Z$, czy jednolity $r \in Z_{N^2}$, ponieważ w obu przypadkach rozkład $[rN \bmod N^2]$ jest taki sam (jak można zweryfikować, patrząc na to, co dzieje się w grupie izomorficznej $Z_N \times Z_N$).

Aby zobaczyć dlaczego to działa, niech $c \equiv (m, r)$ dla dowolnego $r \in \mathbb{Z}_N$. Następnie

$$\begin{aligned} c &\stackrel{\text{zdefiniowane}}{=} [c \cdot \varphi(N) \bmod N]^2 \\ (m, r) \cdot \varphi(N) &= [m \\ &\quad \cdot \varphi(N) \bmod N], [r \cdot \varphi(N) \bmod N] \\ &= [m \cdot \varphi(N) \bmod N], 1. \end{aligned}$$

Zgodnie z Twierdzeniem 13.6(3) oznacza to, że $\hat{c} = (1 + N)[m \cdot \varphi(N) \bmod N] \bmod N^2$. Korzystając z Twierdzenia 13.6(2) wiemy to

$$\hat{c} = (1 + N)[m \cdot \varphi(N) \bmod N] = (1 + [m \cdot \varphi(N) \bmod N] \cdot N) \bmod N^2.$$

Ponieważ $1 + [m \cdot \varphi(N) \bmod N] \cdot N$ jest zawsze mniejsze od N^2 , możemy zrezygnować z mod N^2 na końcu i spójrz na powyższe jako na równość liczb całkowitych. Zatem $\hat{m} (\hat{c} - 1)/N = \frac{\text{def}}{[m \cdot \varphi(N) \bmod N]}$ i w końcu

$$m = [\hat{m} \cdot \varphi(N)]^1 \bmod N,$$

jako wymagane. (Zauważ, że $\varphi(N)$ jest odwracalne modulo N , ponieważ $\gcd(N, \varphi(N)) = 1$.)

Podajemy pełny opis schematu szyfrowania Paillier, a następnie przykład powyższych obliczeń.

BUDOWA 13.11

Niech GenModulus będzie algorytmem działającym w czasie wielomianowym, który na wejściu $1n$ umieszcza (N, p, q) , gdzie $N = pq$ oraz p i q są n -bitowymi liczbami pierwszymi (z wyjątkiem przypadku, gdy prawdopodobieństwo jest znikome w n). Zdefiniuj następujący schemat szyfrowania:

- Gen: na wejściu $1n$ uruchom GenModulus($1n$) aby otrzymać (N, p, q) . Klucz publiczny to N , a klucz prywatny to $N, \varphi(N)$.
- Enc: na wejściu klucz publiczny N i komunikat $m \in \mathbb{Z}_N$ uniform $r \in \mathbb{Z}_N$, Wybierz N i wyprowadź zaszyfrowany tekst

$$\text{do} := [(1 + N)^{m \cdot r} \bmod N^2]$$
- Dec: po wprowadzeniu klucza prywatnego $N, \varphi(N)$ i tekstu zaszyfrowanego c , oblicz

$$m := \frac{[c \cdot \varphi(N) \bmod N^2] - 1}{N} \cdot \varphi(N)^{-1} \bmod N.$$

Schemat szyfrowania Paillier.

Przykład 13.12 Niech

$N = 11 \cdot 17 = 187$ (i tak $N^2 = 34969$) i rozważmy zaszyfrowanie wiadomości $m = 175$, a następnie odszyfrowanie odpowiedniego tekstu zaszyfrowanego. Wybieranie

$r = 83 \quad Z \ 187$, obliczamy szyfrogram

$$c := [(1 + 187)175 \cdot 83187 \bmod 34969] = 23911$$

odpowiadający $(175, 83)$. Aby odszyfrować, zauważ, że $\varphi(N) = 160$. Zatem najpierw obliczamy $\hat{c} := [23911160 \bmod 34969] = 25620$. Odejmowanie 1 i dzielenie przez 187 daje $\hat{m} := (25620 - 1)/187 = 137$; ponieważ $90 = [160 - 1 \bmod 187]$, wiadomość jest odtwarzana jako $m := [137 \cdot 90 \bmod 187] = 175$.

TWIERDZENIE 13.13 Jeśli decyzyjny złożony problem resztowości jest trudny w porównaniu z GenModulus, wówczas schemat szyfrowania Pailliera jest bezpieczny CPA.

DOWÓD Niech Π oznacza schemat szyfrowania Pailliera. Udowodnimy, że Π ma nieroróżnialne szyfrowanie w obecności podsłuchującego; z Twierdzenia 11.6 oznacza to, że jest on bezpieczny pod względem CPA.

Niech A będzie dowolnym probabilistycznym przeciwnikiem wielomianowym w czasie. Rozważmy następujący algorytm ppt D, który próbuje rozwiązać decyzyjny problem rezystywności złożonej w odniesieniu do GenModulus:

Algorytm D:

Algorytm otrzymuje N , y jako dane wejściowe.

- Ustaw $pk = N$ i uruchom A(pk), aby uzyskać dwa komunikaty m_0, m_1 . • Wybierz jednolity bit b i ustaw $c := [(1 + N)mb \cdot y \bmod N^2]$. • Przekaż szyfrogram c A i uzyskaj bit wyjściowy b . Jeśli $b = b$, wyjście 1; w przeciwnym razie wyjście 0.

Przeanalizujmy zachowanie D. Należy rozważyć dwa przypadki:

Przypadek 1: Założymy, że dane wejściowe do D zostały wygenerowane poprzez uruchomienie GenModulus($1n$) do $\bmod N^2$. otrzymać (N, p, q) , wybierając jednostkę $y := r$ [r] (Oznacza to, że y jest jednolitym elementem $\text{Res}(N^2)$). W tym przypadku

$$\text{do} = [(1 + N)mb \cdot r^{N-2} \bmod N^2]$$

dla jednorodnego $r \in Z_{N^2}$. Przypominając, że dystrybucja na $[r^{N-2} \bmod N^2]$ jest tak samo, czy r jest wybrane równomiernie z Z_N lub z Z widzimy, to w tym przypadku Z, widok A, gdy jest uruchamiany jako podprogram przez D, jest identyczny z widokiem A w eksperymencie PubKeav A, $\Pi(n)$. Ponieważ D wyprowadza 1 dokładnie wtedy, gdy wynik b A jest równy b, mamy

$$\Pr[D(N, [r^{N-2} \bmod N^2]) = 1] = \Pr[\text{PubKeav } A, \Pi(n) = 1],$$

gdzie pierwsze prawdopodobieństwo przyjmuje się z eksperymentu jak w definicji 13.10.

Przypadek 2: Założymy, że dane wejściowe do D zostały wygenerowane poprzez uruchomienie GenModulus($1n$) w celu uzyskania (N, p, q) i wybór uniformnego y . Uważamy, że pogląd A

w tym przypadku jest niezależna od bitu b . Wynika to z faktu, że y jest jednolitym elementem grupy Z , a więc zaszyfrowany tekst c jest równomiernie rozłożony w Z^N

(patrz Lemat 11.15), niezależnie od m . Zatem prawdopodobieństwo, że $b = b$ w tym przypadku wynosi dokładnie $\frac{1}{2}$. To jest,

$$\Pr[D(N, r) = 1] = \frac{1}{2},$$

gdzie prawdopodobieństwo przyjmuje się z eksperymentu jak w definicji 13.10.

Łącząc powyższe, widzimy to

$$\begin{aligned} \Pr[D(N, [r^N \bmod N^2]) = 1] &= \Pr[D(N, r) = 1] \\ &= \Pr[\text{PubKeav } A, \Pi(n) = 1] \quad \text{T2.} \end{aligned}$$

Zakładając, że decyzyjny problem resztowości złożonej jest trudny w stosunku do GenModulus, istnieje pomijalna funkcja negl taka, że

$$\Pr[\text{PubKeav } A, \Pi(n) = 1] \xrightarrow{\negl} \text{zaniedb}(n).$$

Zatem $\Pr[\text{PubKeav } A, \Pi(n) = 1] \xrightarrow{\negl} \text{negl}(n)$, kończąc dowód. ■

13.2.3 Szyfrowanie homomorficzne

Schemat szyfrowania Pailliera jest przydatny w wielu ustawieniach, ponieważ jest homomorficzny. Z grubsza homomorficzny schemat szyfrowania umożliwia (pewne) obliczenia na zaszyfrowanych danych, w wyniku czego otrzymuje się zaszyfrowany tekst zawierający zaszyfrowany wynik. W przypadku szyfrowania Pailliera obliczeniem, które można wykonać, jest dodawanie (modułowe). W szczególności napraw klucz publiczny $pk = N$. Wtedy schemat Pailliera ma tę właściwość, że pomnożenie szyfrowania m_1 i szyfrowania m_2 (z mnożeniem wykonanym mod-ulo N^2) daje w wyniku szyfrowanie $[m_1 + m_2 \bmod N]$; to dlatego, że

$$\begin{aligned} &(1 + N) m_1 \cdot r_1^{N^2} \cdot (1 + N) m_2 \cdot r_2^{N^2} \bmod N^2 \\ &= (1 + N) [m_1 + m_2 \bmod N] \cdot (r_1 r_2)^N. \end{aligned}$$

Chociaż możliwość dodawania zaszyfrowanych wartości może nie wydawać się zbyt przydatna, wystarczy do kilku interesujących zastosowań, w tym do głosowania, omówionych poniżej.

Przedstawiamy ogólną definicję, której szczególnym przypadkiem jest szyfrowanie Paillera.

DEFINICJA 13.14 Schemat szyfrowania kluczem publicznym (Gen , Enc , Dec) jest homomorficzny, jeśli dla wszystkich n i wszystkich (pk, sk) wyjść z $\text{Gen}(1n)$ możliwe jest zdefiniowanie grup M , C (w zależności tylko od pk) takie, że:

- Przestrzenią wiadomości jest M , a wszystkie szyfrogramy wysyłane przez Enc_{pk} są elementami C . Dla wygody zapisu M piszemy jako grupę addytywną, a C jako grupę multiplikatywną.

- Dla dowolnego $m_1, m_2 \in M$, dowolnego wyjścia c_1 przez $\text{Encpk}(m_1)$ i dowolnego wyjścia c_2 przez $\text{Encpk}(m_2)$, potwierdza to

$$\text{Pokład}(c_1 \cdot c_2) = m_1 + m_2.$$

Ponadto rozkład na zaszyfrowanych tekstach otrzymanych poprzez zaszyfrowanie m_1 , zaszyfrowanie m_2 , a następnie pomnożenie wyników jest identyczny z rozkładem na zaszyfrowanych tekstach uzyskanych poprzez zaszyfrowanie $m_1 + m_2$.

Ostatnia część definicji zapewnia, że jeśli zostaną wygenerowane szyfrogramy $c_1 = \text{Encpk}(m_1)$ i $c_2 = \text{Encpk}(m_2)$ i obliczony zostanie wynik $c_3 := c_1 \cdot c_2$, to wynikowy szyfrogram c_3 nie będzie zawierał więcej informacji o m_1 lub m_2 niż suma m_3 .

Schemat szyfrowania Pailliera z $pk = N$ jest homomorficzny z $M = ZN$. Nie jest to pierwszy i $C = Z$ przykład homomorficznego schematu szyfrowania, jaki widzieliśmy; Szyfrowanie El Gamala jest również homomorficzne. W szczególności dla klucza publicznego $pk = G, q, g, h$ możemy przyjąć $M = G$ i $C = G \times G$; Następnie

$$g^{y_1}, \quad hy_1 \cdot m_1 \cdot g^{y_2}, \quad hy_2 \cdot m_2 = g^{y_1+y_2}, \quad hy_1+y_2 \cdot m_1 m_2,$$

gdzie mnożenie szyfrogramów odbywa się w oparciu o komponenty. Schemat szyfrowania Goldwassera-Micali, który zobaczymy później, również jest homomorficzny (patrz ćwiczenie 13.10).

Dobrą cechą szyfrowania Pailliera jest to, że jest ono homomorficzne w stosunku do dużej grupy dodatków (mianowicie ZN). Aby zobaczyć zastosowanie tego, rozważ następujący schemat głosowania rozproszonego, w którym wyborcy mogą głosować na „nie” lub „tak”, a celem jest zestawienie liczby głosów na „tak”:

- Organ wyborczy generuje klucz publiczny N w celu szyfrowania Pailliera schemat i publikuje N .
- Niech 0 oznacza „nie”, a 1 oznacza „tak”. Każdy wyborca oddaje swój głos poprzez jego szyfrowanie. Oznacza to, że wyborca i oddaje swój głos v_i obliczając mod N^2 dla jednolitego r_i
- $Z_{ci} := [(1 + N)v_i \cdot r_i] \mod N$.
- Każdy wyborca transmituje swój głos ci . Głosy te są następnie publicznie zatwierdzane agregowane metodą obliczeniową

$$\text{łącznie := } \prod_{i=1}^n ci \mod N^2.$$

- Uprawnienie podaje się ctot. (Zakładamy, że organ nie był w stanie dotychczas obserwować, co się dzieje.) Odszyfrowując go, organ uzyskuje całkowitą liczbę głosów

$$\text{całkowity} \stackrel{\text{def}}{=} \prod_{i=1}^n vi \mod N.$$

Jeśli jest mały (tak, że $vtotal \leq N$), nie ma zawijanego modulo N i $vtotal = \prod_{i=1}^n vi$.

Kluczową cechą powyższego jest to, że żaden wyborca nie dowiaduje się o głosie kogokolwiek innego, a obliczenie sumy można publicznie zweryfikować, jeśli organ ma pewność, że poprawnie obliczy vtot al z ctot. Organ uzyskuje także prawidłową sumę, nie poznać poszczególnych głosów. (Tutaj zakładamy, że władza nie może zobaczyć szyfrogramów wyborców. W sekcji 13.3.3 pokazujemy protokół, w którym głosy są ukrywane przed władzami, nawet jeśli widzą całą komunikację.)

Zakładamy, że wszyscy wyborcy postępują uczciwie (i starają się poznać głosy innych jedynie na podstawie zaobserwowanych informacji); cały obszar badań kryptograficznych poświęcony jest rozwiązywaniu potencjalnych zagrożeń ze strony uczestników, którzy mogą być złośliwi i nie przestrzegać protokołu.

13.3 Udostępnianie sekretów i szyfrowanie progowe

Motywem omówieniem głosowania rozproszonego w poprzedniej sekcji, pokrótko rozważymy bezpieczne (interaktywne) protokoły. Takie protokoły mogą być znacznie bardziej skomplikowane niż podstawowe prymitywy kryptograficzne (np. schematy szyfrowania i podpisu), na których skupialiśmy się do tej pory, zarówno dlatego, że mogą obejmować wiele stron wymieniających kilka rund wiadomości, jak i dlatego, że są przeznaczone do realizacji bardziej złożonych wymagań bezpieczeństwa.

Celem tej sekcji jest przede wszystkim umożliwienie czytelnikowi posmakowania tego fascynującego obszaru i nie podejmuje się próby przedstawienia jej całościowo lub wyczerpująco. Chociaż można wykazać bezpieczeństwo przedstawionych tutaj protokołów (w odniesieniu do odpowiednich definicji), pomijamy formalne definicje, szczegóły i dowody i zamiast tego opieramy się na nieformalnej dyskusji.

13.3.1 Udostępnianie sekretów

Rozważ następujący problem. Dealer posiada tajemnicę – powiedzmy kod wystrzelenia broni nuklearnej – którą chce udostępnić pewnej grupie N użytkowników P_1, \dots, P_N , dając każdemu użytkownikowi udział. Dowolni t użytkownicy powinni być w stanie połączyć swoje udziały i zrekonstruować tajemnicę, ale żadna koalicja składająca się z mniej niż t użytkowników nie powinna uzyskać żadnych informacji o s ze swoich zbiorowych udziałów (poza informacjami, które już o nich posiadałi). Taki mechanizm udostępniania nazywamy progowym schematem udostępniania sekretów (t, N). Taki schemat zapewnia, że s nie zostanie ujawniony bez wystarczającej autoryzacji, a jednocześnie gwarantuje dostępność s w razie potrzeby (ponieważ każdy t użytkownik może je zrekonstruować). Poza bezpośredniem zastosowaniem schematy udostępniania sekretów stanowią również element składowy wielu protokołów kryptograficznych.

Rozważmy proste rozwiązanie dla przypadku $t = N$, zakładając $s \in \{0, 1\}^N$. Dealer wybiera uniformnie losowe $s \in \{0, 1\}^N$. Następnie dla każdego użytkownika i (oznaczanego przez indeks $i = 1, \dots, N$) dealer wykona konstrukcję, w której udział użytkownika P_i wynosi s_i . Ponieważ $s = s_1 \oplus \dots \oplus s_N$, użytkownicy razem mogą odzyskać s . Jednak akcje jakiejkolwiek koalicji

$N - 1$ użytkowników jest (łącznie) jednakowych i niezależnych od s , a zatem nie ujawnia żadnych informacji o s . Jest to jasne, gdy koalicją jest P_1, \dots, P_{N-1} . W ogólnym przypadku, gdy w koalicji są wszyscy oprócz P_j ($j = N$), jest to prawdą, ponieważ $s_1, \dots, s_{j-1}, s_{j+1}, \dots, s_{N-1}$ są jednolite i niezależne od s konstrukcyjnie, oraz

$$s_N = s \quad i < N, i \neq j \text{ i } s_i = s_j;$$

zatem nawet uwarunkowany pewnymi ustalonymi wartościami s i udziałami pozostałych członków koalicji udział s_N użytkownika P_N jest jednolity, ponieważ s_j jest jednolity i niezależny od s .

Możemy to rozszerzyć, aby otrzymać rozwiązanie dla $t < N$. Podstawową ideą jest powtórzenie powyższego schematu dla każdego podzbioru $T \subseteq N$ o rozmiarze t . Oznacza to, że dla każdego takiego podzbioru $T = \{P_{i_1}, \dots, P_{i_t}\}$, wybieramy udziały jednolite $s_{T,i_1}, \dots, s_{T,i_t}$ podlega ograniczeniu, że $\sum_{j=1}^t s_{T,j} = s$, i daje s_{T,i_j} użytkownikowi P_{i_j} . Nietrudno zauważyć, że spełnia to wymagania.

Niestety, to rozszerzenie pierwotnego schematu nie jest efektywne. Każdy użytkownik przechowuje teraz udział $s_{T,i}$ dla każdego podzbioru T , którego jest członkiem. Dla każdego użytkownika istnieją takie podzbiory, które są wykładnicze w N , jeśli $t = N/2$.

Schemat Shamira. Na szczęście można to zrobić znacznie lepiej, korzystając ze schematu udostępniania sekretów wprowadzonego przez Adi Shamira (sławnego z RSA). Schemat ten opiera się na wielomianach² na skończonym ciele F , gdzie F jest tak dobrane, że $s \in F$ i $|F| > N$. (Zobacz Dodatek A.5 zawierający krótkie omówienie cał skończonych.) Zanim opiszemy schemat, krótko omówimy pewne tło związane z wielomianami na ciele F .

Wartość $x \in F$ jest pierwiastkiem wielomianu p , jeśli $p(x) = 0$. Korzystamy z dobrze znanego faktu, że każdy niezerowy wielomian stopnia t nad ciałem ma co najwyżej t pierwiastków. Oznacza to:

WNIOSEK 13.15 Dowolne dwa różne wielomiany stopnia t i q zgadzają się w co najwyżej $t + q$ punktach.

DOWÓD Jeśli nie, to niezerowy wielomian stopnia t $p - q$ miałby więcej niż t pierwiastków. ■

Schemat Shamira opiera się na fakcie, że dla dowolnych t par elementów $(x_1, y_1), \dots, (x_t, y_t) \in F$ (z odrębnością $\{x_i\}$) istnieje unikalny wielomian p stopnia $(t-1)$ taki, że $p(x_i) = y_i$ dla $i = 1, \dots, t$. Możemy to dość łatwo udowodnić.
Fakt, że istnieje taka aplikacja, wykorzystuje standardową interpolację wielomianową.

2A wielomian stopnia t p przez F jest określony przez $p(X) = \sum_{i=0}^t a_i X^i$, a $a_i \in F$. X są zmienną formalną. (Zauważ, że dopuszczałyśmy $a_0 = 0$, więc tak naprawdę mamy na myśli wielomian stopnia co najwyżej t .) Każdy taki wielomian w naturalny sposób definiuje funkcję odwzorowującą F na samą siebie, określoną przez ocenę wielomianu na jego wejściu.

Szczegółowo: dla $i = 1, \dots, t$, zdefiniuj wielomian stopnia $(t - 1)$.

$$\delta_i(x) \stackrel{\text{def}}{=} \frac{\sum_{j=1, j \neq i}^{t-1} (x - x_j)}{(x_i - x_j)}.$$

Zauważ, że $\delta_i(x_j) = 0$ dla dowolnego $j = i$ oraz $\delta_i(x_i) = 1$. Zatem $p(X) \delta_i(X) \stackrel{\text{def}}{=} p_i(X)$ jest wielomianem stopnia $(t - 1)$ z $p(x_i) = y_i$ dla $1 \leq i \leq t$. (Zauważamy, że w rzeczywistości pokazuje to, że żądanego wielomianu p można znaleźć efektywnie.) Wyjątkowość wynika z Wniosku 13.15.

Opiszemy teraz schemat udostępniania sekretów z programem (t, N) Shamira. Niech F będzie ciałem skończonym zawierającym dziedzinę możliwych tajemnic, i gdzie $|F| > N$. Niech $x_1, \dots, x_N \in F$ będą odrębnymi, niezerowymi elementami, które są stałe i publicznie znane. (Takie elementy istnieją, ponieważ $|F| > N$.) Schemat działa w następujący sposób:

Dzielenie się: Mając sekret $s \in F$, krupier wybiera jednolity $a_1, \dots, a_{t-1} \in F$ definiując wielomian $p(X)$ stopnia $(t - 1) \stackrel{\text{def}}{=} s + \sum_{i=1}^{t-1} a_i X^i$. Jest to jednorodność i wielomian o stałym członie s . Udział użytkownika P_i wynosi $s_i := p(x_i) \in F$.

Rekonstrukcja: Powiedzmy t użytkownikom P_1, \dots, P_t basen ich udziały s_1, \dots, s_{t-1} , usiądź. Używając interpolacji wielomianowej, obliczają unikalny wielomian stopnia $(t - 1)$ p , dla którego $p(x_j) = s_j$ dla $1 \leq j \leq t$. Sekretem jest $p(0)$.

Jest oczywiste, że rekonstrukcja działa, ponieważ $p = p \circ p(0) = s$.

Pozostaje pokazać, że dowolny użytkownik $t - 1$ nie dowie się nic o tajemnicach s ze swoich udziałów. Aby zachować symetrię, wystarczy wziąć pod uwagę udziały użytkowników P_1, \dots, C_{t-1} . Twierdzimy, że dla dowolnego sekretu s udziały s_1, \dots, s_{t-1} są (łącznie) jednolite. Ponieważ krupier wybiera a_1, \dots, a_{t-1} równomiernie, wynika to z tego, jeśli pokażemy, że istnieje zgodność jeden do jednego między wielomianem p wybranym przez dealera a udziałami s_1, \dots, s_{t-1} . Jest to jednak bezpośrednia konsekwencja Wniosku 13.15.

13.3.2 Sprawdzalne udostępnianie sekretów

Do tej pory rozważaliśmy ataki pasywne, w których $t - 1$ użytkowników może próbować wykorzystać swoje udziały w celu uzyskania informacji o tajemnicach. Ale możemy również martwić się aktywnym, złośliwym zachowaniem. Istnieją dwie odrębne kwestie: po pierwsze, skorumpowany dealer może udostępniać użytkownikom niespójne udziały, tj. w taki sposób, że odzyskiwane są różne sekrety w zależności od tego, który użytkownik łączy swoje udziały. Po drugie, w fazie rekonstrukcji złośliwy użytkownik może przedstawić inny udział niż ten przekazany mu przez dealera i w ten sposób wpłynąć na odzyskany sekret. (Chociaż można temu zaradzić, zlecając dealerowi podpisanie akcji, nie działa to, gdy sam dealer może być nieuczciwy). Sprawdzalne schematy udostępniania sekretów (VSS) zapobiegają obu tym atakom.

Bardziej formalnie, pozwalamy, aby dowolny użytkownik $t - 1$ był skorumpowany i zmawiał się ze sobą i ewentualnie z krupierem. Wymagamy (1a) na końcu

faza udostępniania, sekret jest zdefiniowany w taki sposób, że każda kolekcja zawierająca t nieuskodzonych użytkowników (niezależnie od tego, czy ta kolekcja zawiera również niektórych uszkodzonych użytkowników), pomyślnie odzyska je w fazie rekonstrukcji; ponadto (1b) jeśli handlarz jest uczciwy, wówczas s odpowiada tajemnicy handlarza. Ponadto (2) gdy dealer jest uczciwy, wówczas, tak jak poprzednio, skorumpowani użytkownicy t - 1 nie dowiadują się nic o tajemnicy swoich udziałów ani żadnych informacji publicznych publikowanych przez dealera. Ponieważ chcemy, aby było t nieskorumpowanych użytkowników, nawet jeśli t - 1 użytkowników jest uszkodzonych, wymagamy $N = t + (t - 1) > 2(t - 1)$; innymi słowy, zakładamy, że większość użytkowników pozostaje nieuskodzona.

Opisujemy schemat VSS autorstwa Feldmana, który opiera się na algorytmie G, w stosunku do którego problem logarytmu dyskretnego jest trudny. Dla uproszczenia opisujemy to w modelu losowej wycieczki i niech H oznacza funkcję, która ma być modelowana jako losowa wyciecznia. Zakładamy również, że pewne zaufane parametry (G, q, g), generowane za pomocą G(1n), są publikowane z wyprzedzeniem, gdzie q jest liczbą pierwszą, a więc Zq jest ciałem. Na koniec zakładamy, że wszyscy użytkownicy mają dostęp do kanału transmisji, tak że wiadomość nadawana przez dowolnego użytkownika jest słyszana przez wszystkich.

Faza udostępniania obejmuje teraz N użytkowników prowadzących interaktywny protokół z dealerem, który przebiega w następujący sposób:

1. Aby podzielić się sekretem s, krupier wybiera jednolite $a_0 \in Z_q$, a następnie dzieli się a_0 jak w schemacie Shamira. Oznacza to, że krupier wybiera jednolite $a_i x_i$. The dealer wysyła udział si := $p(i) = \sum_{j=0}^{t-1} a_i \cdot j$ do użytkownika P_i .
Ponadto dealer publicznie udostępnia wartości $A_0 := g$ At-1 := g^{a_0} , ..., oraz „zamaskowany sekret” $c := H(a_0) \in S$.

2. Każdy użytkownik P_i sprawdza, czy jego udział si spełnia

$$si \stackrel{?}{=} t \prod_{j=0}^{t-1} (A_j)^{a_j} g^{ja_j}. \quad (13.3)$$

Jeśli nie, P_i publicznie ogłasza skargę.

Pamiętaj, że jeśli sprzedawca jest uczciwy, to tak

$$\prod_{j=0}^{t-1} (A_j)^{a_j} = \prod_{j=0}^{t-1} (g a_j)^{a_j} = g^{\sum_{j=0}^{t-1} a_j \cdot j} = g^{p(i)} \text{ si } = g = ,$$

i tak żaden uczciwy użytkownik nie będzie narzekał. Ponieważ jest co najwyżej t-1 skorumpowanych użytkowników, może być co najwyżej t-1 skarg, jeśli krupier jest uczciwy.

3. Jeżeli więcej niż t-1 użytkowników złoży skargę, krupier zostaje zdyskwalifikowany, a protokół zostaje przerwany. W przeciwnym razie dealer odpowiada na skargę P_i , nadając komunikat si. Jeżeli udział ten nie spełnia równania (13.3) (lub jeśli dealer w ogóle odmówi odpowiedzi na reklamację), dealer zostaje zdyskwalifikowany, a protokół zostaje przerwany. W przeciwnym razie P_i wykorzystuje wartość rozgłoszoną (a nie wartość otrzymaną w pierwszej rundzie) jako swój udział.

³Zauważ, że teraz ustawiamy $x_i = i$, co jest w porządku, ponieważ używamy pola Z_q .

Załóżmy, że w fazie rekonstrukcji grupa użytkowników (w tym co najmniej t nieuskodzonych użytkowników) łączy swoje udziały. Udział si dostarczony przez użytkownika P_i jest odrzucony, jeśli nie spełnia równania (13.3). Spośród pozostałych udziałów dowolne t z nich służy do odzyskania a_0 dokładnie tak, jak w schemacie Shamira. Pierwotny sekret jest następnie obliczany jako $s := c \cdot H(a_0)$.

Obecnie twierdzimy, że protokół ten spełnia pożądane wymagania bezpieczeństwa. Najpierw pokazujemy, że zakładając, że dealer nie jest zdyskwalifikowany, wartość odzyskana w fazie odbudowy jest jednoznacznie określona na podstawie informacji publicznych; w szczególności odzyskana wartość wynosi $c \cdot H(\log g A_0)$. (W połączeniu z faktem, że uczciwy krupier nigdy nie zostaje zdyskwalifikowany, dowodzi to, że warunki (1a) i (1b) są spełnione.) Zdefiniuj $a_i := \log g A_i$ dla $0 \leq i \leq t-1$; $\{a_i\}$ nie może być efektywnie obliczone, jeśli problem logarytmu dyskretnego jest trudny, ale nadal są dobrze zdefiniowane. Zdefiniuj wielomian $p(X)$ wniesiony przez część P_i , który nie jest odrzucony w fazie rekonstrukcji, musi spełniać Równanie (13.3), a zatem spełnia $\sum_{j=0}^{t-1} a_i X^j = p(i)$. Wynika z tego, że niezależnie od tego, jakie udziały zastosujemy, strony zrekonstruują wielomian p , obliczą $a_0 = p(0)$, a następnie odzyskają $s = c \cdot H(a_0)$.

Można również wykazać, że warunek (2) obowiązuje dla przeciwników ograniczonych obliczeniowo, jeśli problem logarytmu dyskretnego jest trudny dla G . (W przeciwieństwie do schematu udostępniania sekretów Shamira, tajemnica nie jest tu już bezwarunkowa. Bezwarunkowo bezpieczna Schematy VSS są możliwe, ale wykraczają poza zakres naszego omówienia.) Intuicyjnie dzieje się tak, ponieważ sekret s jest maskowany przez losową wartość $H(a_0)$ i informacje przekazywane dowolnym $t - 1$ użytkownikom w fazie udostępniania — mianowicie, ich udziały i wartości publiczne $\{A_i\}$ — ujawnia tylko g , z którego trudno wyliczyć a_0 . Tę intuicję można uczynić rygorystyczną, ale tutaj tego nie robimy.

a_0 ,

13.3.3 Szyfrowanie progowe i głosowanie elektroniczne

W sekcji 13.2.3 wprowadziliśmy pojęcie homomorficznych schematów szyfrowania i jako przykład podaliśmy schemat szyfrowania Pailliera. Tutaj pokazujemy inny schemat szyfrowania homomorficznego, który jest wariantem szyfrowania El Gamala. W szczególności, mając klucz publiczny $pk = G, q, g, h$ jak w zwykłym szyfrowaniu El Gamala, szyfrujemy teraz wiadomość $m \in \mathbb{Z}_q$ poprzez ustalenie $M := g^m$, wybranie jednolitego $y \in \mathbb{Z}_q$ i wysłanie zaszyfrowanego tekstu $c := g^y \cdot M$. Aby odszyfrować, odbiorca odzyskuje M jak w standardowym deszyfrowaniu El Gamala, a następnie oblicza $m := \log_g M$. Chociaż nie jest to efektywne, jeśli m pochodzi z dużej domeny, jeśli m pochodzi z małej domeny — tak jak to będzie w naszej aplikacji — wówczas odbiorca będzie mógł efektywnie obliczyć $\log M$ za pomocą wyszukiwania wyczerpującego. Zaletą tego schematu wariantów jest to, że jest on homomorficzny pod względem dodawania w \mathbb{Z}_q . To jest,

$$g^{y_1}, \quad hy_1 \cdot g^{m_1} \quad y_2 \cdot g, \quad hy_2 \cdot g^{m_2} = rz \quad g^{y_1+y_2}, \quad hy_1+y_2 \cdot g^{m_1+m_2}.$$

Przypomnijmy, że podstawowe podejście do głosowania elektronicznego z wykorzystaniem homomorficznych en-

szyfrowanie polega na tym, że każdy wyborca szyfruje swój głos $v_i \in \{0, 1\}$, aby otrzymać zaszyfrowany tekst c_i . Kiedy już wszyscy zagłosują, szyfrogramy są mnożone w celu uzyskania zaszyfrowania sumy $v_{total} = v_1 + v_2 + \dots + v_n$. (Wartość q jest w praktyce wystarczająco dużą, aby nie wystąpiło żadne zawijane modulo q). Ponieważ $0 \leq v_{total} \leq n$, gdzie oznacza całkowitą liczbę wyborców, organ posiadający klucz prywatny może skutecznie odszyfrować c_{total} , końcowy tekst zaszyfrowany i odzyskać v_{total} .

Wadą tego podejścia jest to, że ufa się temu organowi, że zarówno (poprawnie) odszyfruje końcowy tekst zaszyfrowany, jak i nie odszyfruje żadnego z zaszyfrowanych tekstów poszczególnych wyborców. (W sekcji 13.2.3 założyliśmy, że organ nie może zobaczyć szyfrogramów poszczególnych wyborców.) Zamiast tego możemy woleć dystrybucję zaufania pomiędzy zestawem N organów, tak aby każdy zestaw t organów był w stanie wspólnie odszyfrować uzgodniony tekst zaszyfrowany (zapewnia to dostępność, nawet jeśli niektóre organy nie działają lub nie chcą pomóc w odszyfrowaniu), ale żadna kolekcja $t - 1$ organów nie jest w stanie samodzielnie odszyfrować żadnego tekstu zaszyfrowanego (zapewnia to prywatność, o ile mniej niż t organów jest uszkodzonych).

Na pierwszy rzut oka może się wydawać, że udostępnianie sekretów rozwiązuje problem. Jeśli udostępnimy klucz prywatny pomiędzy N autorytetami, wówczas żaden zestaw autorytetów $t-1$ nie pozna klucza prywatnego i nie będzie mógł go odszyfrować. Z drugiej strony, dowolne władze mogą połączyć swoje udziały, odzyskać klucz prywatny, a następnie odszyfrować dowolny żądany tekst zaszyfrowany.

Krótką myśl pokazuje, że to nie do końca działa. Jeśli władze zrekonstruują klucz prywatny w celu odszyfrowania jakiegoś zaszyfrowanego tekstu, to w ramach tego procesu wszystkie władze poznają klucz prywatny! Zatem później dowolny organ może samodzielnie odszyfrować dowolny wybrany przez siebie szyfrogram.

Zamiast tego potrzebujemy zmodyfikowanego podejścia, w którym „sekret” (mianowicie klucz prywatny) nigdy nie jest rekonstruowany w sposób wyraźny, ale w sposób dorozumiany rekonstruowany jest tylko na tyle, aby umożliwić odszyfrowanie jednego, uzgodnionego tekstu zaszyfrowanego. Możemy to osiągnąć w konkretnym przypadku szyfrowania El Gamal w następujący sposób. Ustal klucz publiczny $pk = G, q, g, h$ i niech $x \in \mathbb{Z}_q$ będzie kluczem prywatnym, tj. $= h^x$. Każdemu i -temu organowi przydzielany jest udział $x_i \in \mathbb{Z}_q$ dokładnie tak, jak w schemacie udostępniania sekretów Shamira. Oznacza to, że wybierany jest wielomian p o jednolitym stopniu ($t - 1$), gdzie $p(0) = x$ i podaje się i -ty autorytet $x_i := p(i)$. (Zakładamy, że jest to zaufany dealer, który zna x i bezpiecznie usuwa go po udostępnieniu. Możliwe jest całkowite wyeliminowanie dealera, ale wykracza to poza nasz obecny zakres.)

Teraz powiedzmy t autorytetom i_1, \dots, i_t , chce wspólnie odszyfrować szyfrogram x_{ij} . Odnośnie-
 c_1, c_2 . Aby to zrobić, autorytet i_j najpierw publikuje wartość $w_{ij} := c_1$ wywołanie z $j=1$
 poprzedniej sekcji, że istnieją publicznie obliczalne wielomiany $\{\delta_j(X)\}$ (które zależą od tożsamości tych t autorytetów) takie, że $p(X) = \sum_{j=1}^t \delta_j(X) \cdot x_{ij}$. Ustawienie $\delta_j = \delta_j(0)$, widzimy, że istnieje publicznie $j=1 \dots t$ δ_j
 Dowolny autorytet i_j otrzymuje $w_{ij} = \delta_j(0) = \sum_{j=1}^t \delta_j(0) \cdot x_{ij} = \sum_{j=1}^t \delta_j \cdot x_{ij}$.

$$M := \frac{c_2}{\prod_{j=1}^t \delta_j}.$$

(W razie potrzeby każdy z nich może następnie obliczyć $\log M$.) Aby sprawdzić, czy to poprawnie odtworzy komunikat, powiedz $c1 = g^y$ i $c2 = h^y$

$$\sum_{j=1}^T w_j^{x_j} = \sum_{j=1}^T c_1^{x_j} \delta_j = \text{do}_1 \sum_{j=1}^T x_j \delta_j = \text{do}_1^{p(0)} = \text{do}_1^X,$$

a więc

$$M \stackrel{\text{def}}{=} \frac{c_2}{\prod_{j=1}^T w_j^{x_j}} = \frac{h^y \cdot M}{g^y \cdot M} = \frac{(g^y) \cdot M}{y(g^y) \cdot x} = M.$$

Należy zauważać, że dowolny zbiór $t - 1$ skorumpowanych władz nie dowiaduje się niczego o kluczu prywatnym x ze swoich udziałów. Co więcej, można wykazać, że z procesu deszyfrowania nie uczą się niczego poza odzyskaną wartością M .

Złośliwi (aktywni) przeciwnicy. Powyższe podejście zakłada, że wszystkie osoby odszyfrowujące jakiś zaszyfrowany tekst zachowują się prawidłowo. (Jeśli tak nie jest, każdemu z nich łatwo byłoby spowodować nieprawidłowy wynik, publikując dowolną wartość w_j .) Zakładamy również, że wyborcy zachowują się uczciwie i szyfrujemy głos 0 lub 1. (Zauważ, że wyborca może nieuczciwie wpływać na wynik wyborów, szyfrując dużą lub ujemną wartość.) Potencjalnemu złośliwemu zachowaniu tego rodzaju można zapobiec, stosując techniki wykraczające poza zakres tej książki.

13.4 Schemat szyfrowania Goldwassera-Micali

Zanim przedstawimy schemat szyfrowania Goldwassera – Micali, musimy lepiej zrozumieć reszty kwadratowe. Najpierw zbadamy łatwiejszy przypadek reszt kwadratowych modulo a liczba pierwsza p , a następnie przyjrzymy się nieco bardziej skomplikowanemu przypadkowi reszt kwadratowych modulo a złożony N .

W całej tej sekcji p i q oznaczają nieparzyste liczby pierwsze, a $N = pq$ oznacza iloczyn dwóch różnych, nieparzystych liczb pierwszych.

13.4.1 Reszty kwadratowe modulo a prime

W grupie G element $y \in G$ jest resztą kwadratową, jeśli istnieje $x \in G$ z $x^2 \equiv y \pmod{p}$. W tym przypadku x nazywamy pierwiastkiem kwadratowym z y . Element, który nie jest resztą kwadratową, nazywany jest nieresztą kwadratową. W grupie abelowej zbiór reszt kwadratowych tworzy podgrupę.

W konkretnym przypadku $Z \bmod p$, że y jest resztą kwadratową, jeśli istnieje $2x \equiv y \pmod{p}$. Zaczynamy od łatwej obserwacji.

TWIERDZENIE 13.16 Niech $p > 2$ będzie liczbą pierwszą. Każda reszta kwadratowa w $Z \bmod p$ ma dokładnie dwa pierwiastki kwadratowe.

DOWÓD Niech $y \in Z$ takie, p być resztą kwadratową. Wtedy istnieje $x \in Z$ takie, że $x^2 = y \pmod{p}$. Jasne, $(-x)^2 = x^2 = y \pmod{p}$. Ponadto $x \equiv -x \pmod{p}$: jeśli $x \equiv x \pmod{p}$ to $2x \equiv 0 \pmod{p}$, co implikuje $p \mid 2x$. Ponieważ p jest liczbą pierwszą, oznaczałoby to, że albo $p \mid 2$ (co jest niemożliwe, ponieważ $p > 2$) lub $p \mid x$ (co jest niemożliwe, ponieważ $0 < x < p$). Zatem $[x \pmod{p}] \cap [-x \pmod{p}]$ są odrębnymi elementami Z_p , i y ma co najmniej dwa pierwiastki kwadratowe. Niech $x \in Z_p$ takie, że $x^2 = y = (x)$ będzie pierwiastkiem kwadratowym z y . Wtedy $x \equiv 0 \pmod{p}$. Otrzymujemy rozkład lewej strony

$$(x - x)(x + x) = 0 \pmod{p},$$

tak więc (zgodnie z Twierdzeniem 8.3) albo $p \mid (x - x)$ albo $p \mid (x + x)$. W pierwszym przypadku $x \equiv x \pmod{p}$, a w drugim przypadku $x \equiv -x \pmod{p}$, co pokazuje, że y rzeczywiście ma tylko $\{\pm x \pmod{p}\}$ jako pierwiastki kwadratowe. ■

Niech kwadrat : $Z \subseteq \mathbb{Z}$ będzie funkcją $sq(x) \pmod{p}$.⁴ Powyższe pokazuje, że sq jest funkcją dwa do jednego, gdy $p > 2$ jest liczbą pierwszą. To natychmiast oznacza, że dokładnie połowa elementów Z to reszty kwadratowe. Zbiór reszt kwadratowych modulo p oznaczamy przez QR_p , a zbiór nieresztów kwadratowych przez QN_p . Właśnie widzieliśmy, że dla $p > 2$ liczba pierwsza

$$|QR_p| = |QN_p| = \frac{\frac{|Z|}{2}}{2} = \frac{p-1}{2}.$$

Zdefiniuj $Jp(x)$, symbol Jacobiego $x \pmod{p}$, w następujący sposób.⁴ Niech $p > 2$ będzie liczbą pierwszą, a $x \in Z \pmod{p}$. Następnie

$$Jp(x) \stackrel{\text{def}}{=} \begin{cases} +1 & \text{j jeśli } x \text{ jest resztą kwadratową modulo } p \\ -1 & \text{j jeśli } x \text{ nie jest resztą kwadratową modulo } p. \end{cases}$$

Zapis można w naturalny sposób rozszerzyć dla dowolnego x względnie pierwszego do p , ustawiając $Jp(x) = Jp([x \pmod{p}])$.

Czy możemy scharakteryzować reszty kwadratowe w $Z \pmod{p}$? Zaczynamy od faktu, że jest $Z \cong Z_{p-1}$ to grupa cykliczna rzędu $p-1$ (patrz Twierdzenie 8.56). Niech g będzie generatorem Z_{p-1} . To znaczy że

$$Z_{p-1} = \{ \text{przyp. } 0, g^1, g^{2, \dots}, G^{\frac{p-1}{2}, r_z} \} = \{ g^{p-1}, g^{p-1-2}, \dots, g^{p-1-2} \}$$

(przypomnijmy, że p jest nieparzyste, więc $p-1$ jest parzyste). Podnosząc do kwadratu każdy element tej listy i redukując modulo $p-1$ w wykładniku (por. Wniosek 8.15) otrzymujemy listę wszystkich reszt kwadratowych w Z

$$QR_p = \{ g^0, g^2, g^4, \dots, g^{p-3}, g^0, g^2, \dots, g^{p-3} \}.$$

⁴Dla p prime, $Jp(x)$ jest czasami nazywane symbolem Legendre'a x i oznaczane przez $Lp(x)$; wybraliśmy naszą notację tak, aby była zgodna z notacją wprowadzoną później.

Każda reszta kwadratowa pojawia się dwukrotnie na tej liście. Dlatego reszty kwadratowe w Z_p są dokładnie tymi elementami, które można zapisać jako g przy $i \in \{0, \dots, p-2\}$ parzysta liczba całkowita.

Powyższa charakterystyka prowadzi do prostego sposobu obliczenia, czy Jacobi jest resztą symbol i w ten sposób stwierdzić, czy element $x \in Z_p$ kwadratową, czy nie.

TWIERDZENIE 13.17 Niech $p > 2$ będzie liczbą pierwszą. Wtedy $J_p(x) = x^{\frac{p-1}{2}} \mod s. 2$

DOWÓD Niech g będzie dowolnym generatorem Z_p . Jeśli x jest resztą kwadratową dla naszej wcześniejszej dyskusji wynika, że $x = g^i$ jakiejś parzystej liczby całkowitej i . Zapisując $i = 2j$ z liczbą całkowitą, mamy wtedy

$$\frac{p-1}{2}x = \text{sol} \quad 2j^{\frac{p-1}{2}} g^{(p-1)j} = \text{sol} \quad p^{-1} \text{jot} = 1 \text{ mod } p, =$$

i tak $x^{\frac{p-1}{2}} = +1 = J_p(x) \mod p$, jak twierdzono.

Z drugiej strony, jeśli x nie jest resztą kwadratową, to $x = g^i$ liczba całkowita tj. Zapisując $i = 2j+1$, gdzie j jest liczbą całkowitą

dla jakiegoś dziwnego

$$\frac{p-1}{2}x = g \quad 2j+1^{\frac{p-1}{2}} = g \quad 2j^{\frac{p-1}{2}} \cdot j^{\frac{p-1}{2}} = 1 \cdot j^{\frac{p-1}{2}} = g^{\frac{p-1}{2}} \mod s. 2$$

Teraz,

$$\frac{p-1}{2}g^2 = 1 \mod p, = g$$

i tak $G^{\frac{p-1}{2}} = \pm 1 \mod p$, ponieważ $[\pm 1 \mod p]$ to dwa pierwiastki kwadratowe z 1 (por. Twierdzenie 13.16). Ponieważ g jest generatorem, ma rząd $p-1$ i tak $\frac{p-1}{2} = 1 \mod s. 2$ Wynika z tego, że $x^{\frac{p-1}{2}} = 1 = J_p(x) \mod p$.

■

Twierdzenie 13.17 podaje bezpośrednio algorytm czasu wielomianowego (por. Algo-jest resztą rytm 13.18) do sprawdzania, czy element $x \in Z_p$ kwadratową.

ALGORYTM 13.18 Decydowanie

resztości kwadratowej modulo a prime

Dane wejściowe: Liczba pierwsza p ; element $x \in Z_p$

Wynik: $J_p(x)$ (lub równoważnie, czy x jest resztą kwadratową, czy nieresztą kwadratową)

$$b := x^{\frac{p-1}{2}} \mod p \text{ jeśli } b =$$

1 zwróć „resztę kwadratową” w przeciwnym razie zwróć „nieresztę kwadratową”

Zakończymy tę sekcję zwracając uwagę na ciekawą multiplikatywną właściwość kwadratu reszty i nie-resztki modulo p .

TWIERDZENIE 13.19 Niech $p > 2$ będzie liczbą pierwszą, a $x, y \in \mathbb{Z}$

str Następnie

$$J_p(xy) = J_p(x) \cdot J_p(y).$$

DOWÓD Korzystając z poprzedniego twierdzenia,

$$J_p(xy) = (xy)^{\frac{p-1}{2}} = x^{\frac{p-1}{2}} \cdot y^{\frac{p-1}{2}} = J_p(x) \cdot J_p(y) \text{ mod } p.$$

Ponieważ $J_p(xy) \cdot J_p(x) \cdot J_p(y) = \pm 1$, równość obowiązuje również w przypadku liczb całkowitych. ■DODATEK 13.20 Niech $p > 2$ będzie liczbą pierwszą i powiedzmy $x, x \in QR_p$ i $y, y \in QN_p$. Następnie:

1. $[xx \text{ mod } p] \in QR_p$.
2. $[yy \text{ mod } p] \in QR_p$.
3. $[xy \text{ mod } p] \in QN_p$.

13.4.2 Reszty kwadratowe Modulo a Composite

Skupmy się teraz na resztach kwadratowych w grupie $Z_N = pq$. Scharakteryzowaniami. Gdzie reszt kwadratowych modulo N jest łatwe, jeśli zastosujemy wyniki z poprzedniej sekcji w połączeniu z chińskim twierdzeniem o resztach. Przypomnijmy, że chińskie twierdzenie o resztach mówi, że Z i niech $y \in (yp, yq)$ oznaczają zgodność gwarantowaną $\begin{cases} N \\ Z \end{cases} \equiv \begin{cases} p \times Z \\ q \end{cases}$, przez twierdzenie (tj. $yp = [y \text{ mod } p]$ i $yq = [y \text{ mod } q]$). Kluczową obserwacją jest:

TWIERDZENIE 13.21 Niech $N = pq$ z p, q różnymi liczbami pierwszymi i $y \in Z_N$ z $y \in (yp, yq)$. Wtedy y jest resztą kwadratową modulo N wtedy i tylko wtedy, gdy yp jest resztą kwadratową modulo p i yq jest resztą kwadratową modulo q .

DOWÓD Jeśli y jest resztą kwadratową modulo N , to z definicji istnieje $= y \text{ mod } N$. Niech x takie, że $x^2 \equiv (xp, xq) \text{ mod } N$. Wtedy istnieje $x \in Z_N$

$$(yp, yq) \quad y = x^2 \quad (xp, xq) \quad x^2 = ([x^2 \text{ mod } p], [x^2 \text{ mod } q]),$$

gdzie $(xp, xq) \in Z_q^2$ jest po prostu kwadratem elementu (xp, xq) w grupie Z_p . W ten sposób $p \times Z_q^2$ to pokazaliśmy

$$yp = x^2 \text{ mod } p \text{ i } yq = x^2 \text{ mod } q \quad (13,4)$$

oraz yp, yq są resztami kwadratowymi (w odniesieniu do odpowiednich modułów).

I odwrotnie, jeśli $y \equiv (yp, yq) \pmod{N}$ i yp, yq są resztami kwadratowymi modulo p i q takimi, że równanie (13.4) odpowiednio, to istnieją zasady $xp \equiv Z \pmod{p}$ i $xq \equiv Z \pmod{q}$ jest takie, że $x \equiv (xp, xq) \pmod{N}$. Odwrócenie Niech $x \in \mathbb{Z}_N$ pokazuje powyższych kroków
że x jest pierwiastkiem kwadratowym z y modulo N . ■

Powyższe twierdzenie charakteryzuje reszty kwadratowe modulo N . Dokładne zbadanie dowodu prowadzi do kolejnej ważnej obserwacji: każda reszta kwadratowa $y \in \mathbb{Z}$ ma dokładnie cztery pierwiastki kwadratowe. Aby to zobaczyć, niech $y \equiv (yp, yq)$ będzie resztą kwadratową modulo N i niech xp, xq będą pierwiastkami kwadratowymi odpowiednio yp i yq modulo p i q . Następnie cztery pierwiastki kwadratowe z y są dane przez elementy w \mathbb{Z}_N odpowiadającej

$$(xp, xq), (-xp, xq), (xp, -xq), (-xp, -xq). \quad (13.5)$$

Każdy z nich jest pierwiastkiem kwadratowym z y , ponieważ

$$\begin{aligned} (\pm xp, \pm xq)^2 &= [(\pm xp)^2 \pmod{p}], [(\pm xq)^2 \pmod{q}] \\ &= ([x^2 \pmod{p}], [x^2 \pmod{q}]) = (yp, yq) \equiv y \end{aligned}$$

(gdzie ponownie zapis (\cdot, \cdot) odnosi się do kwadratu w grupie $\mathbb{Z}_p \times \mathbb{Z}_q$).

Chińskie twierdzenie o resztach gwarantuje, że cztery elementy równania (13.5) odpowiadają różnym elementom \mathbb{Z} modulo p (i \mathbb{Z} modulo q). Ponieważ xp i xq są unikalne podobnie dla xq i xp modulo q).

Przykład 13.22

Rozważmy \mathbb{Z}_{15} . Zgodność wynikająca z chińskim twierdzeniem o resztach jest 15 zestawiona przykładzie 8.25). Element 4 to reszta kwadratowa modulo 15 z pierwiastkiem kwadratowym z 2. Ponieważ $2 \equiv (2, 2)$, pozostałe pierwiastki kwadratowe z 4 są dane przez

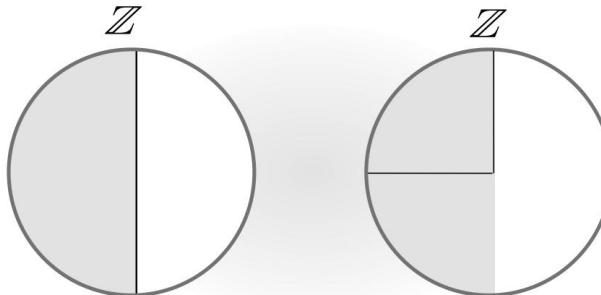
- $2, [2 \pmod{3}] = (2, 1) \equiv 7;$
- $[2 \pmod{5}], 2 = (3, 2) \equiv 8; I$
- $[2 \pmod{5}], [-2 \pmod{3}] = (3, 1) \equiv 13.$

Można sprawdzić, że $7^2 = 8^2 = 13^2 = 4 \pmod{15}$.

Niech QRN oznacza zbiór reszt kwadratowych modulo N . Ponieważ podniesienie kwadratu modulo N jest funkcją cztery do jednego, widzimy, że dokładnie $1/4$ elementów \mathbb{Z} to \mathbb{Z}_N są resztami kwadratowymi. Alternatywnie, moglibyśmy zauważyc, że skoro $y \in \mathbb{Z}$ jest resztą kwadratową wtedy i tylko wtedy, gdy yp, yq są resztami kwadratowymi, istnieje zgodność jeden do jednego pomiędzy QRN i $\mathbb{Z}_p \times \mathbb{Z}_q$. Zatem ułamek reszt kwadratowych modulo N wynosi

$$\frac{|QRN|}{|\mathbb{Z}_N|} = \frac{|QRp| \cdot |QRq|}{|\mathbb{Z}_N|} = \frac{\frac{p-1}{2} \cdot \frac{q-1}{2}}{(p-1)(q-1)} = \frac{1}{4},$$

zgodnie z powyższym.

RYSUNEK 13.1: Struktura Z_p i Z_N .

W poprzedniej sekcji zdefiniowaliśmy symbol Jacobiego $Jp(x)$ dla liczby pierwszej $p > 2$. Rozszerzamy definicję na przypadek N będący iloczynem różnych, nieparzystych liczb pierwszych p i q w następujący sposób. Dla dowolnego x względnie pierwszego do $N = pq$,

$$\begin{aligned} JN(x) &\stackrel{\text{def}}{=} Jp(x) \cdot Jq(x) \\ &= Jp([x \bmod p]) \cdot Jq([x \bmod q]). \end{aligned}$$

Definiujemy \mathbb{Z}_N^{+1} jako zbiór elementów w \mathbb{Z}_N posiadających symbol Jacobiego $+1$, oraz zdefiniuj JN analogicznie.

Z Twierdzenia 13.21 wiemy, że jeśli x jest resztą kwadratową modulo N , to $[x \bmod p]$ i $[x \bmod q]$ są resztami kwadratowymi odpowiednio modulo p i q ; to znaczy $Jp(x) = Jq(x) = +1$. Zatem $JN(x) = +1$ i widzimy, że:

Jeśli x jest resztą kwadratową modulo N , to $JN(x) = +1$.

Jednakże $JN(x) = +1$ może również wystąpić, gdy $Jp(x) = Jq(x) = -1$, to znaczy, gdy zarówno $[x \bmod p]$, jak i $[x \bmod q]$ nie są resztami kwadratowymi modulo p i q (więc x nie jest resztą kwadratową modulo N). Okazuje się to przydatne w przypadku schematu szyfrowania Goldwassera – Micali, dlatego dla zbioru elementów tego typu wprowadzamy notację QN_{R+1} . To jest,

$$QN_{R+1} \stackrel{\text{def}}{=} x \in \mathbb{Z}_N \quad x \text{ nie jest resztą kwadratową modulo } N, \text{ ale } JN(x) = +1.$$

Można teraz łatwo udowodnić, co następuje (patrz rysunek 13.1):

TWIERDZENIE 13.23 Niech $N = pq$ z p, q różnymi, nieparzystymi liczbami pierwszymi. Następnie:

1. Dokładnie połowa elementów \mathbb{Z}_N są w JN^{+1} .
2. QN_{R+1} zawarty jest w JN^{+1} .
3. Dokładnie połowa elementów JN_{R+1} są w QN_{R+1} (druga połowa jest w JN_{R+1}).

DOWÓD Wiemy, że $JN(x) = +1$ jeśli $Jp(x) = Jq(x) = +1$ lub $Jp(x) = Jq(x) = -1$. Wiemy również (z poprzedniego rozdziału), że dokładnie połowa elementów Z ma symbol Jacobiego $+1$, a połowa ma symbol Jacobiego -1 w naturalnym

1 (i podobnie dla drogi Z , mamy zatem

$$\begin{aligned} J_N^{+1} &= \left| \left\{ p^{+1} \times q^{+1} \mid + \right\} \cup \left| \left\{ p^{-1} \times q^{-1} \mid - \right\} \right| \right| \\ &= \left| \left\{ p^{+1} \mid \cdot \right\} \cup \left| \left\{ q^{+1} \mid \cdot \right\} \right| + \left| \left\{ p^{-1} \mid \cdot \right\} \cup \left| \left\{ q^{-1} \mid \cdot \right\} \right| \right| \\ &= \frac{(p-1)(q-1)}{2} + \frac{(p-1)(q-1)}{2} = \frac{\varphi(N)}{2}. \end{aligned}$$

Więc $J_N^{+1} = |Z_N|/2$, udowadniając, że połowa elementów Z_N są w J_N^{+1} .

Zauważliśmy wcześniej, że wszystkie reszty kwadratowe modulo N mają Jacobiego symbol $+1$, pokazując, że $QRN \subset J_N^{+1}$.

Ponieważ $x \in QRN$ wtedy i tylko wtedy, gdy $Jp(x) = Jq(x) = +1$, mamy

$$|QRN| = \left| \left\{ p^{+1} \times q^{+1} \mid + \right\} \right| = 2 \frac{(p-1)(q-1)}{2} = \frac{\varphi(N)}{4},$$

i tak $|QRN| = J_N^{+1}$ połowa Z_N /2. Ponieważ QRN jest podzbiorem J_N^{+1} , to tego dowodzi elementów J_N^{+1} są one w QRN . ■

Następne dwa wyniki są analogami Twierdzenia 13.19 i Wniosku 13.20.

TWIERDZENIE 13.24 Niech $N = pq$ będzie iloczynem różnych, nieparzystych liczb oraz $x, y \in Z$ pierwszych, N . Wtedy $JN(xy) = JN(x) \cdot JN(y)$.

DOWÓD Używając definicji $JN(\cdot)$ i Twierdzenia 13.19:

$$\begin{aligned} JN(xy) &= Jp(xy) \cdot Jq(xy) = Jp(x) \cdot Jp(y) \cdot Jq(x) \cdot Jq(y) \\ &= Jp(x) \cdot Jq(x) \cdot Jp(y) \cdot Jq(y) = JN(x) \cdot JN(y). \end{aligned}$$



DODATEK 13.25 Niech $N = pq$ będzie iloczynem różnych, nieparzystych liczb pierwszych i powiedzmy $x, x \in QRN$ i $y, y \in QN R+1 N$. Następnie:

$$1. [xx \bmod N] \in QRN.$$

$$2. [yy \bmod N] \in QN.$$

$$3. [xy \bmod N] \in QN R+1 N.$$

DOWÓD Dowodzimy twierdzenia końcowego; dowody pozostałych są podobne. Ponieważ , mamy $Jp(x) = Jq(x) = +1$. Ponieważ $y \in QN R+1$ i $x \in QRN$ mamy N , $Jp(y) = Jq(y) = -1$. Korzystając z Twierdzenia 13.19,

$$Jp(xy) = Jp(x) \cdot Jp(y) = -1 \text{ i } Jq(xy) = Jq(x) \cdot Jq(y) = -1,$$

i tak $JN(xy) = +1$. Ale xy nie jest resztą kwadratową modulo N , ponieważ $Jp(xy) = -1$, zatem $[xy \bmod p]$ nie jest resztą kwadratową modulo p . Dochodzimy do wniosku, że $xy \in QN R+1 \bmod N$.

W przeciwnieństwie do Wniosku 13.20 nie jest prawdą, że $y, y \in QN RN$ implikuje $yy \in QRN$. (Zamiast tego, jak wskazano w następstwie, jest to gwarantowane tylko wtedy, gdy $y, y \in QN R+1$.) Na przykład moglibyśmy mieć $Jp(y) = +1, Jq(y) = -1$ i $Jp(y) = -1, Jq(y) = +1$, więc $Jp(yy) = Jq(yy) = -1$ i yy nie jest resztą kwadratową, mimo że $JN(yy) = +1$.

13.4.3 Założenie rezydualności kwadratowej

W podrozdziale 13.4.1 pokazaliśmy skuteczny algorytm decydowania, czy dane wejściowe x są resztą kwadratową modulo p a liczba pierwsza p . Czy możemy dostosować algorytm do pracy modulo z liczbą złożoną N ? Twierdzenie 13.21 daje łatwe rozwiązanie tego problemu, pod warunkiem, że znana jest faktoryzacja N . Zobacz Algorytm 13.26.

ALGORYTM 13.26

Decydowanie resztości kwadratowej modulo złożenie znanej faktoryzacji

Dane wejściowe: Złożone $N = pq$; czynniki p i q ; element $x \in Z \bmod N$

Wynik: decyzja, czy $x \in QRN$ obliczyć $Jp(x)$ i $Jq(x)$,

jeśli $Jp(x) = Jq(x) = +1$ zwróć

„resztę kwadratową”, w przeciwnym razie zwróć

„nieresztę kwadratową”

(jak zawsze zakładamy, że czynniki N są różnymi liczbami pierwszymi nieparzystymi.) Prosta modyfikacja powyższego algorytmu pozwala obliczyć $JN(x)$, gdy znana jest faktoryzacja N .

Gdy jednak rozkład na czynniki N nie jest znany, nie jest znany algorytm czasu wielomianowego decydujący, czy dany x jest resztą kwadratową modulo N , czy nie. Co nieco zaskakujące, znany jest algorytm czasu wielomianowego umożliwiający obliczanie $JN(x)$ bez rozkładu na czynniki N . (Chociaż sam algorytm nie jest aż tak skomplikowany, jego dowód poprawności wykracza poza zakres tej książki i dlatego nie przedstawiają w ogóle algorytmu. Zainteresowany czytelnik może zapoznać się z literaturą wymienioną na stronie

koniec tego rozdziału.) Prowadzi to do częściowego testu resztości kwadratowej: jeśli dla danych wejściowych x utrzymuje się, że $JN(x) = -1$, to x nie może być resztą kwadratową. (Patrz Twierdzenie 13.23.) Ten test nic nie mówi, gdy $JN(x) = +1$ i nie jest znany żaden wielomianowy algorytm określający rezydualność kwadratową w tym przypadku (jest to lepsze niż zgadywanie losowe).

Sformalizujemy teraz założenie, że problem ten jest trudny. Niech GenModulus będzie algorytmem działającym w czasie wielomianowym, który na wejściu $1n$ daje wyniki (N, p, q) , gdzie $N = pq$, a p i q są n -bitowymi liczbami pierwszymi, z wyjątkiem prawdopodobieństwa zaniedbywalnego w n .

DEFINICJA 13.27 Mówimy, że określenie resztości kwadratowej jest trudne w odniesieniu do GenModulus, jeśli dla wszystkich probabilistycznych algorytmów wielomianowych D istnieje funkcja pomijalna taka, że

$$\Pr[D(N, qr) = 1] = \Pr[D(N, qnr) = 1] = \text{negl}(n),$$

gdzie w każdym przypadku prawdopodobieństwa są przejmowane w eksperymencie, w którym GenModulus($1n$) jest uruchamiany w celu otrzymania (N, p, q) , qr wybiera się jednolicie z QRN , a qnr wybiera się jednolicie z $QN R+1$

$$N.$$

W powyższym przypadku istotne jest, aby qnr zostało wybrane spośród $QN R+1$ zamiast $QN RN$; gdyby qnr zostało wybrane spośród $QN RN$, to z prawdopodobieństwem $2/3$ byłoby tak, że $JN(x) = -1$, a zatem odróżnienie qnr od jednolitej reszty kwadratowej byłoby łatwe. (Przypomnijmy, że $JN(x)$ można efektywnie obliczyć nawet bez rozkładu na czynniki N .)

Założenie dotyczące rezydualności kwadratowej jest po prostu założeniem, że istnieje GenModulus, względem którego określenie resztości kwadratowej jest trudne. Łatwo zauważać, że jeśli określenie resztości kwadratowej jest trudne w odniesieniu do GenModulus, to rozkład na czynniki musi być również trudny w odniesieniu do GenModulus.

13.4.4 Schemat szyfrowania Goldwassera-Micali

W poprzedniej sekcji natychmiast zasugerowano schemat szyfrowania klucza publicznego dla komunikatów jednobitowych w oparciu o założenie rezydualności kwadratowej:

- Klucz publiczny to moduł N , a klucz prywatny to jego faktoryzacja.
- Aby zaszyfrować „0”, wyślij jednolitą resztę kwadratową; aby zaszyfrować „1”, wyślij jednolitą kwadratową nieresztą z symbolem Jacobiego +1.
- Odbiorca może odszyfrować zaszyfrowany tekst c swoim kluczem prywatnym, korzystając z rozkładu na czynniki N , aby zdecydować, czy c jest resztą kwadratową, czy nie.

Bezpieczeństwo CPA tego schematu wynika niemal trywialnie z trudności problemu resztowości kwadratowej, sformalizowanej w Definicji 13.27.

W powyższym opisie brakuje określenia, w jaki sposób nadawca, nie znając rozkładu N , może wybrać mundur

element QRN (do szyfrowania 0) lub jednolity element QN R+1 (do szyfrowania N). To pierwsze jest łatwe, drugie wymaga pewnej pomysłowości.

Wybór jednolitej reszty kwadratowej. Wybór elementu jednolitego y QRN jest prosty: wystarczy wybrać element jednolity x Z (patrz Dodatek B.2.5) i ustawić mod N. Oczywiście y 2 lata := x QRN . Fakt, że y jest równomiernie rozłożone w QRN wynika z faktu, że podniesienie do kwadratu modulo N jest funkcją 4 do 1 N . Bardziej szczegółowo, ustalmy dowolny ^y QRN (patrz sekcja 13.4.2) i że x jest wybrane jednolicie z Z i obliczmy prawdopodobieństwo, że y = ^y po powyższej procedurze. Oznacz cztery pierwiastki kwadratowe z ^y przez ±x, ±x^ . Następnie:

$$\begin{aligned} \Pr[y = \hat{y}] &= \Pr[x \text{ to pierwiastek kwadratowy z } \hat{y}] \\ &= \Pr[x \in \{\pm x, \pm x^2\}] \\ &= \frac{4}{|Z_N|} = \frac{1}{|QRN|}. \end{aligned}$$

Ponieważ powyższe obowiązuje dla każdego ^y , widzimy, że y jest rozłożone jednoformalnie w QRN .

Wybór elementu jednolitego QN R+1 N . Generalnie nie wiadomo, jak wybrać element jednorodny QN R+1, jeśli nie jest znana rozkład na czynniki N.

W obecnym kontekście ratuje nas to, że odbiorca może pomóc, włączając pewne informacje do klucza publicznego. Konkretnie modyfikujemy schemat tak, aby odbiorca dodatkowo wybierał jednolite z QN R+1 i zawierał z jako część swojego klucza publicznego. (Odbiorca może to łatwo zrobić, ponieważ zna rozkład na czynniki N; patrz ćwiczenie 13.6.) Nadawca może wybrać element jednolity y QN R+1 wybierając uniform x Z (jak wyżej) i ustawiając mod N] Z Wniosku 13.25 wynika, że y QN R+1 N . My N

y := [z · x]^2 zostaw to jako ćwiczenie, aby pokazać, że y jest równomiernie rozłożone w QN R+1 . Nie N ; My wykorzystuj tego faktu bezpośrednio w poniższym dowodzie bezpieczeństwa.

Pełny opis schematu szyfrowania Goldwassera – Micali, realizującego powyższe pomysły, podajemy w Construction 13.28.

TWIERDZENIE 13.29 Jeśli problem resztości kwadratowej jest trudny w porównaniu z GenModulus, wówczas schemat szyfrowania Goldwassera-Micali jest zabezpieczony CPA.

DOWÓD Niech Π oznacza schemat szyfrowania Goldwassera – Micali. Udowodnimy, że Π ma nieroróżnialne szyfrowanie w obecności podsłuchującego; z Twierdzenia 11.6 oznacza to, że jest on bezpieczny pod względem CPA.

Niech A będzie dowolnym probabilistycznym przeciwnikiem wielomianowym w czasie. Rozważmy następującego przeciwnika ppt D, który próbuje rozwiązać problem resztowości kwadratowej w odniesieniu do GenModulus:

Algorytm D:

Algorytm otrzymuje na wejściu N i z, a jego celem jest określenie, czy z QRN czy z QN R+1 N .

BUDOWA 13.28

Niech GenModulus będzie jak zwykle. Skonstruuj schemat szyfrowania klucza publicznego w następujący sposób:

- Gen: na wejściu $1n$, uruchom GenModulus($1n$), aby uzyskać (N, p, q) i N . wybierz uniform z $QN R+1$. Klucz publiczny to $pk = N$, z i kluczem prywatnym jest $sk = p, q$.
 - Enc: na wejściu klucz publiczny $pk = N$, z i wiadomość $m \in \{0, 1\}$, a na wyjściu wybierz mundur $x \in Z_N$ zaszyfrowany tekst
- $$c := [z^m \cdot 2 \cdot x \mod N].$$
- Dec: po wprowadzeniu klucza prywatnego sk i tekstu zaszyfrowanego c , ustal, czy c jest resztą kwadratową modulo N , korzystając np. z algorytmu 13.26. Jeśli tak, wyjście 0; w przeciwnym razie wyjście 1.

Schemat szyfrowania Goldwassera – Micali.

- Ustaw $pk = N$, z i uruchom $A(pk)$, aby otrzymać dwa jednobitowe komunikaty m_0, m_1 .
- Wybierz jednolity bit b i jednolity $x \in Z$ $c := [z \cdot m_b \mod N]$, a następnie ustaw $\mod N$. • Przełącz zaszyfrowany tekst c A, który z kolei wyśle bit b . Jeśli $b = b$, wyjście 1; w przeciwnym razie wyjście 0.

Przeanalizujmy zachowanie D. Należy rozważyć dwa przypadki:

Przypadek 1: Założymy, że dane wejściowe do D zostały wygenerowane poprzez uruchomienie GenModulus($1n$) w celu uzyskania (N, p, q) , a następnie wybranie jednolitego z $QN R+1$. Następnie D uruchamia A na kluczu publicznym skonstruowanym dokładnie tak jak w Π i widzimy, że w tym przypadku widok A uruchomiony jako podprogram przez D jest identyczny z widokiem A w eksperymencji PubKeav A, $\Pi(n)$. Ponieważ D wyprowadza 1 dokładnie wtedy, gdy wynik b A jest równy b, mamy

$$\Pr[D(N, qnr) = 1] = \Pr[PubKeav A, \Pi(n) = 1], \text{ gdzie}$$

qnr reprezentuje element jednorodny $QN R+1$ N jak w definicji 13.27.

Przypadek 2: Założymy, że dane wejściowe do D zostały wygenerowane poprzez uruchomienie GenModulus($1n$) w celu uzyskania (N, p, q) , a następnie wybranie jednolitego z QRN . Twierdzimy, że pogląd A w tym przypadku jest niezależny od bitu b . Aby to zobaczyć, zauważ, że zaszyfrowany tekst c podany A jest jednolitą resztą kwadratową, niezależnie od tego, czy zaszyfrowane jest 0 czy 1:

- Gdy zaszyfrowane jest 0, $c = [x \text{ jest } 2 \mod N]$ dla jednolitego $x \in Z_N$, i tak C jednolitą resztą kwadratową.
- Gdy 1 jest zaszyfrowana, $c = [z \cdot x \mod N]$ dla jednolitego $x \in Z_N$. Niech $x \stackrel{\text{def}}{=} [x^2 \mod N]$ i zauważ, że x jest elementem o rozkładzie jednostajnym

z grupy QRN . Ponieważ z QRN wnioskujemy,, możemy zastosować Lemat 11.15 do że c jest równomiernie rozłożone również w QRN.

Ponieważ pogląd A jest niezależny od b, prawdopodobieństwo, że $b = b$ w tym przypadku wynosi dokładnie $\frac{1}{2}$. To jest,

$$\Pr[D(N, qr) = 1] = \frac{1}{2},$$

gdzie qr reprezentuje jednolity element QRN zgodnie z definicją 13.27.

Zatem,

$$\Pr[D(N, qr) = 1] = \Pr[D(N, qnr) = 1] = \Pr[\text{PubKeav } A, \Pi(n) = 1]$$

T2.

Zakładając, że problem resztości kwadratowej jest trudny w stosunku do GenModulus, istnieje pomijalna funkcja negl taka, że

$$\varepsilon(n) = \frac{1}{2} + \text{zaniedbywanie}(n);$$

zatem $\varepsilon(n) = \frac{1}{2} + \text{zaniedbywanie}(n)$. To kończy dowód. ■

13.5 Schemat szyfrowania Rabina

Jak wspomniano na początku tego rozdziału, schemat szyfrowania Rabina jest atrakcyjny, ponieważ jego bezpieczeństwo jest równoznaczne z założeniem, że faktoring jest trudny. Analogiczny wynik nie jest znany w przypadku szyfrowania opartego na RSA, a problem RSA może być potencjalnie łatwiejszy niż faktoring. (To samo dotyczy schematu szyfrowania Goldwassera – Micaliego i możliwe jest, że określenie resztości kwadratowej modulo N jest łatwiejsze niż rozkład na czynniki N.)

Co ciekawe, schemat szyfrowania Rabina jest (przynajmniej powierzchownie) bardzo podobny do schematu szyfrowania RSA, ale ma tę zaletę, że opiera się na potencjalnie słabszym założeniu. Wydaje się, że szerzej stosowane RSA niż poprzednie wynika bardziej z czynników historycznych niż technicznych; omówimy to szerzej na końcu tej sekcji.

Zaczniemy od kilku wstępów na temat obliczania modułowych pierwiastków kwadratowych. Następnie wprowadzamy permutację z zapadnią, którą można bezpośrednio opierć na założeniu, że faktoring jest trudny. Następnie uzyskuje się schemat szyfrowania Rabina (lub przynajmniej jedną jego instancję), stosując wyniki z sekcji 13.1. W całej tej sekcji będziemy nadal pozwalać, aby p i q oznaczały nieparzyste liczby pierwsze, a $N = pq$ oznaczały iloczyn dwóch różnych, nieparzystych liczb pierwszych.

13.5.1 Obliczanie modułowych pierwiastków kwadratowych

Schemat szyfrowania Rabina wymaga od odbiornika obliczenia modułowych pierwiastków kwadratowych, dlatego w tej sekcji zbadamy złożoność algorytmiczną

ten problem. Najpierw pokażemy efektywny algorytm obliczania pierwiastków kwadratowych modulo a liczba pierwsza p , a następnie rozszerzymy ten algorytm, aby umożliwić obliczenie pierwiastków kwadratowych modulo a złożonego N o znanej faktoryzacji. Czytelnik chcący na wiarę przyjąć istnienie tych algorytmów może przejść do następnej sekcji, w której pokazujemy, że obliczenie pierwiastków kwadratowych modulo złożonego N o nieznanej faktoryzacji jest równoważne rozkładowi na czynniki N .

Niech p będzie nieparzystą liczbą pierwszą. Obliczanie pierwiastków kwadratowych modulo p jest stosunkowo proste, gdy $p = 3 \pmod{4}$, ale znacznie bardziej skomplikowane, gdy $p = 1 \pmod{4}$. (W przypadku schematu szyfrowania Rabina wystarczy nam łatwiejszy przypadek, jak przedstawiono w sekcji 13.5.3; uwzględniamy drugi przypadek dla kompletności.) W obu przypadkach pokazujemy, jak obliczyć jeden z pierwiastków kwadratowych reszty kwadratowej a . Zauważ, że jeśli x jest jednym z pierwiastków kwadratowych z a , to $[x \pmod{p}]$ jest drugim.

Najpierw zajmiemy się łatwiejszym przypadkiem. Powiedzmy $p = 3 \pmod{4}$, co oznacza, że możemy zapisać $p = 4i + 3$ dla pewnej liczby całkowitej i . Ponieważ $a \pmod{p}$ jest resztą kwadratową, mamy $J_p(a) = 1 = \frac{p+1}{2} \pmod{p}$ (patrz Twierdzenie 13.17). Mnożąc obie strony przez a otrzymujemy

$$= za \quad \frac{p-1}{2} \cdot 2a = a^{2i+2} = za^{i+1} \pmod{p},$$

i tak $a^{i+1} = a^{\frac{p+1}{2}} \pmod{p}$ jest pierwiastkiem kwadratowym z a . Oznacza to, że otrzymujemy kwadrat pierwiastek modulo p po prostu obliczając $x := [a^{\frac{p+1}{4}} \pmod{p}]$.

Ważne jest powyżej, aby $(p+1)/2$ było parzyste, ponieważ zapewnia to, że $(p+1)/4 \pmod{p}$ jest liczbą całkowitą (jest to konieczne, aby przypomnieć, $\frac{p+1}{2}$ będzie dobrze zdefiniowane; że wykładnik musi być liczbą całkowitą). To podejście nie powiedzie się, gdy $p = 1 \pmod{4}$, w którym to przypadku $p+1$ jest liczbą całkowitą, która nie jest podzielna przez 4.

Gdy $p = 1 \pmod{4}$ postępujemy nieco inaczej. Kierując się powyższym podejściem, możemy mieć nadzieję, że znajdziemy nieparzystą liczbę całkowitą r , dla której to obowiązuje $A^r = 1 \pmod{p}$. Następnie, jak wyżej, a $\frac{r+1}{2} \pmod{p}$ i $a^{\frac{r+1}{2}} \pmod{p}$ będzie pierwiastkiem kwadratowym z a , gdzie $(r+1)/2$ jest liczbą całkowitą. Chociaż nie będziemy w stanie tego zrobić, możemy zrobić coś równie dobrego: znajdziemy nieparzystą liczbę całkowitą r wraz z elementem $b \in \mathbb{Z}_p^*$ i parzystą liczbę całkowitą r taką, że

$$A^r \cdot B^r = 1 \pmod{p}.$$

Wtedy $a^{\frac{r+1}{2}} \cdot b^{\frac{r+1}{2}} \pmod{p}$ wynosi $(r+1)/2$ i $r+1 \pmod{2}$ są $\frac{r}{2} \pmod{p}$ jest pierwiastkiem kwadratowym z a (z liczbami całkowitymi).

Opiszemy teraz ogólne podejście do znajdowania r , b i r , przy czym m są liczbami całkowitymi nieruchomości. Niech $\frac{p-1}{2} = 2 \cdot m$ gdzie $p \equiv 1 \pmod{4}$ i m nieparzyste. Ponieważ $a \pmod{p}$ jest resztą kwadratową, wiemy o tym

$$a^m \equiv 1 \pmod{p} \quad \text{czyli} \quad a^{\frac{p-1}{2}} \equiv 1 \pmod{p}. \quad (13.6)$$

Oznacza to, że $a = a^m \equiv 1 \pmod{p}$ jest pierwiastkiem kwadratowym z 1. Pierwiastki kwadratowe $2 \cdot m$ modulo p wynoszą $\pm 1 \pmod{p}$, więc $a^{\frac{p-1}{2}} \equiv \pm 1 \pmod{p}$.

⁵Liczby całkowite i m można łatwo obliczyć, usuwając współczynniki 2 z $(p-1)/2$.

znajdują się w tej samej sytuacji, co w równaniu (13.6), z tą różnicą, że wykładownik a jest teraz podzielny przez mniejszą potęgę 2. Jest to postęp we właściwym kierunku: jeśli uda nam się dojść do punktu, w którym wykładownik a nie jest podzielny przez dowolną potegę 2 (tak jak miałooby to miejsce w tym przypadku, gdyby $a = 1$), wówczas wykładownik a jest nieparzysty i możemy obliczyć pierwiastek kwadratowy, jak omówiliśmy wcześniej. Podamy przykład $2 \equiv 1 \pmod{p}$ i za chwilę omówimy, jak postępować w przypadku, gdy $a \equiv 1 \pmod{p}$.

Przykład 13.30

Weźmy $p = 29$ i $a = 7$. Ponieważ 7 jest resztą kwadratową modulo 29, mamy $14^2 \equiv 1 \pmod{29}$ i wiemy, że $77 \equiv 1 \pmod{29}$ to pierwiastek kwadratowy z 1. W rzeczywistości

$$77 \equiv 1 \pmod{29},$$

a wykładownik 7 jest nieparzysty. Zatem $7(7+1)/2 = 74 \equiv 23 \pmod{29}$ to pierwiastek kwadratowy z 7 modulo 29.

Podsumowując dotychczasowy algorytm: zaczynamy od $a^{2m} \equiv 1 \pmod{p}$ i wyciągamy współczynniki 2 z wykładownika, aż wydarzy się jedna z dwóch rzeczy: albo

$a^m \equiv 1 \pmod{p}$ lub $a = 2^m \pmod{p}$ dla niektórych $m < \frac{p-1}{2}$. W pierwszym przypadku, ponieważ m jest nieparzyste, możemy od razu obliczyć pierwiastek kwadratowy z a, jak w przykładzie 13.30. W drugim przypadku „przywrócimy” +1 po prawej stronie równania, mnożąc każdą stronę równania przez $-1 \pmod{p}$. Jednakże, jak motywowaliśmy na początku tej dyskusji, chcemy to osiągnąć poprzez pomnożenie lewej strony równania przez jakiś element b podniesiony do parzystej potęgi. Jeśli mamy kwadratową nieresztę b $\equiv Z \pmod{p}$, jest to łatwe:

$$\text{ponieważ } b^{\frac{p-1}{2}} \equiv 1 \pmod{p}, \text{ mamy } p,$$

$$2m \cdot ba \equiv (-1)(-1) \equiv +1 \pmod{p}.$$

W ten sposób możemy postępować jak poprzednio, biorąc pierwiastek kwadratowy z lewej strony, aby zmniejszyć największą potegę 2, dzieląc wykładownik a i mnożąc $2m$ (w razie potrzeby),

tak aby prawa strona zawsze wynosiła +1. Zauważ, że wykładownik b przez b jest zawsze podzielny przez większą potegę 2 niż wykładownik a (więc rzeczywiście możemy wyciągnąć pierwiastek kwadratowy, dzieląc przez 2 w obu wykładownikach).

Kontynuujemy wykonywanie tych kroków, aż wykładownik a będzie nieparzysty, a następnie możemy obliczyć pierwiastek kwadratowy z a, jak opisano wcześniej. Pseudokod tego algorytmu, który daje inny sposób patrzenia na to, co się dzieje, podano poniżej w Algorytmie 13.31. Można sprawdzić, że algorytm działa w czasie wielomianowym, mając kwadratową nieresztę b, ponieważ liczba iteracji pętli wewnętrznej wynosi = O(log p).

Jedną z kwestii, którą jeszcze nie poruszyliśmy, jest przede wszystkim to, jak znaleźć b. W rzeczywistości nie jest znany żaden deterministyczny algorytm czasu wielomianowego do znajdowania kwadratowego modulo p nieresztowego. Na szczęście łatwo jest znaleźć kwadratową nieresztę probabilistycznie: po prostu wybieraj jednolite elementy Z aż do

ALGORYTM 13.31 Obliczanie pierwiastków kwadratowych modulo a prime

Wejście: Liczba pierwsza p ; reszta kwadratowa $a \in \mathbb{Z}_p$

Wynik: pierwiastek kwadratowy z a

przypadek $p = 3 \bmod 4$:

return $\sqrt[4]{a \bmod p}$

[przypadek $p = 1 \bmod 4$:

niech b będzie kwadratowym nieresztowym modulo

p obliczonym i m nieparzystym z $2 \cdot m = r := 2 \cdot \frac{p-1}{2}$

$m, r := 0$ dla $i = 1 \dots$ do 1 { //

utrzymuj niezmienny

$ar := r/2, r := r/2$ jeśli $a = 1 \bmod \frac{r}{2} \cdot B^{\frac{r}{2}} = 1 \bmod str$

$pr := r + 2 \cdot m$

$\frac{r}{2} \cdot B^{\frac{r}{2}}$

} // teraz $r = m, r$ jest parzyste i $a 2 \cdot b \frac{r}{2} \cdot B^{\frac{r}{2}} = 1 \bmod str$
powrót $a \frac{r+1}{2} \frac{r}{2} \bmod s. 2$

znaleziono kwadratową nieresztę. Działa to, ponieważ dokładnie połowa elementów \mathbb{Z} to piereszyt kwadratowe i ponieważ znany jest wielomianowy algorytm wyznaczania reszty kwadratowej modulo liczby pierwszej (dowody obu tych twierdzeń znajdują się w rozdziale 13.4.1). Oznacza to, że pokazany przez nas algorytm jest w rzeczywistości losowy, gdy $p = 1 \bmod 4$; deterministyczny algorytm czasu wielomianowego do obliczania pierwiastków kwadratowych w tym przypadku nie jest znany.

Przykład 13.32

Rozważamy tutaj „najgorszy przypadek”, gdy pierwiastek kwadratowy zawsze daje -1 .

Niech $a \in \mathbb{Z}$ będzie elementem, którego pierwiastek kwadratowy chcemy obliczyć; niech str

$b \in \mathbb{Z}_p$ być nieresztą kwadratową; i niech $p-1 = 2 \cdot m$ gdzie m jest nieparzyste.

mamy $a = 1 \bmod p$. Ponieważ $2,2m$ pierwiastek kwadratowy z $1 = za^{2,2m} \bmod p$

wynosi ± 1 , oznacza to, że w najgorszym przypadku $2,2m$, $a = \pm 1 \bmod p$; zakładając, że aby $a = 1 \bmod p$. Zatem mnożymy przez $b^{p-1} = b^{2,2m} \bmod p$

$$a^{2,2m} \bmod p = 1 \bmod p$$

W drugim kroku zauważamy, że $b^{2,2m} \bmod p$ jest pierwiastkiem kwadratowym z zakladając najgorszy przypadek, mamy $b^{2,2m} \bmod p = 1 \bmod p$. Mnożenie zatem $2,2m$ na b do „poprawienia”, co daje

$$b^{2,2m} \bmod p = 1 \bmod p$$

W trzecim kroku, biorąc pierwiastki kwadratowe i zakładając najgorszy przypadek (jak

powyżej otrzymujemy $A^M \cdot B^{2m} \cdot u_r^{2 \cdot 2m} = 1 \pmod{p}$; mnożąc przez „poprawkę współczynnik 2^{3m} ” b otrzymujemy

$$A^M \cdot b \cdot b^{2 \cdot 2m \cdot 2 \cdot 3m \cdot 2m} = 1 \pmod{p}$$

Jesteśmy teraz tam, gdzie chcemy być. Aby zakończyć algorytm,помнóż obie strony przez a, aby otrzymać

$$A^{M+1} \cdot b = \frac{2m+22m+23m}{a \pmod{\text{str.}}}$$

Ponieważ m jest nieparzyste, $(m+1)/2$ jest liczbą całkowitą i pierwiastkiem kwadratowym z a.

$$\xrightarrow{m+1} b^{m+2m+2} 2m \pmod{p} \text{ to } a$$

Przykład 13.33

Tutaj opracowujemy konkretny przykład. Niech $p = 17$, $a = 2$ i $b = 3$. Zauważ, że tutaj $(p-1)/2 = 23$ i $m = 1$.

Zaczynamy od $22 = 1^3 \pmod{17}$. Zatem $22 = 1^2 \pmod{17}$ powinna być równa $\pm 1 \pmod{17}$; przez daje obliczenia, $22 = 1 \pmod{17}$. Mnożenie przez 32 $22 \cdot 3 = 1 \pmod{17}$. jest

Kontynuując, wiemy, że $22 \cdot 3$ do $\pm 1^{22}$ pierwiastkiem kwadratowym z 1, więc musi być $\pmod{17}$; obliczenie daje $22 \cdot 3$ potrzebne $2^{22} \pmod{17}$ równe = 1 mod 17. Zatem nie ma składnika korygującego tutaj.

Dzieląc wykładniki ponownie o połowę, stwierdzamy, $2^2 = 1 \pmod{17}$. Mamy teraz = 2 že $2 \cdot 3$ prawie gotowe: mnożenie obu stron przez 2 daje $22 \cdot 3 = 2^2 \pmod{17}$ i tak $2 \cdot 3 = 6 \pmod{17}$ to pierwiastek kwadratowy z 2.

Obliczanie pierwiastków kwadratowych Modulo N

Nietrudno zauważyc, że pokazany przez nas algorytm do obliczania pierwiastków kwadratowych modulo a prime można łatwo rozszerzyć na przypadek obliczania pierwiastków kwadratowych modulo a złożony $N = pq$ o znanej faktoryzacji. W szczególności niech a będzie resztą kwadratową z (ap, aq) zgodnie z chińskim twierdzeniem o reszcie.

Obliczanie pierwiastków kwadratowych, odpowiednio, xp , xq z ap , aq modulo p i q daje pierwiastek kwadratowy (xp, xq) z a (patrz rozdział 13.4.2). Biorąc pod uwagę xp i xq , reprezentację x odpowiadającą (xp, xq) można odzyskać w sposób omówiony w podrozdziale 8.1.5. Oznacza to, że aby obliczyć pierwiastek kwadratowy z modulo, liczbę całkowitą $N = pq$ znanej faktoryzacji:

- Oblicz $ap := [a \pmod{p}]$ i $aq := [a \pmod{q}]$.
- Używając Algorytmu 13.31, oblicz pierwiastek kwadratowy xp z ap modulo p i pierwiastek kwadratowy xq z aq modulo q.
- Konwertuj z reprezentacji (xp, xq) $\begin{matrix} Z \\ p \end{matrix} \times \begin{matrix} Z \\ q \end{matrix}$ do $x \in \begin{matrix} Z \\ N \end{matrix}$

Algorytm można łatwo zmodyfikować tak, aby zwracał wszystkie cztery pierwiastki kwadratowe z a.

13.5.2 Permutacja zapadni oparta na faktoringu

Widzieliśmy, że obliczanie pierwiastków kwadratowych modulo N można przeprowadzić w czasie wielomianowym, jeśli znana jest faktoryzacja N. Pokazujemy tutaj, że dla kontrastu obliczenie pierwiastka kwadratowego modulo złożonego N o nieznanej faktoryzacji jest tak samo trudne jak rozkład na czynniki N.

Bardziej formalnie, niech GenModulus będzie algorytmem działającym w czasie wielomianowym, który na wejściu $1n$ wyprowadza (N, p, q) , gdzie $N = pq$ i p i q są n -bitowymi liczbami pierwszymi, z wyjątkiem prawdopodobieństwa zaniedbywalnego w n . Rozważmy następujący eksperyment dla danego algorytmu A i parametru n :

Eksperyment obliczeniowy z pierwiastkiem kwadratowym SQRA,GenModulus(n):

1. Uruchom GenModulus($1n$), aby uzyskać dane wyjściowe N, p, q .
2. Wybierz mundur $y \in QRN$.
3. Dane jest $A(N, y)$ i wyniki $x \in Z$. Wynik N . Eksperymentu definiuje się jako 1, jeśli $x^2 \equiv y \pmod{N}$, i 0 w przeciwnym razie.

DEFINICJA 13.34 Mówimy, że obliczanie pierwiastków kwadratowych jest trudne w porównaniu z GenModulus, jeśli dla wszystkich probabilistycznych algorytmów wielomianowych A w czasie istnieje pomijalna funkcja negl taka, że

$$\Pr[\text{SQRA,GenModulus}(n) = 1] = \text{negl}(n).$$

Łatwo zauważyc, że jeśli obliczanie pierwiastków kwadratowych jest trudne w stosunku do GenModulus , to rozkład na czynniki musi być trudny również w odniesieniu do GenModulus : jeśli moduły N wyjściowe GenModulus można łatwo rozłożyć na czynniki, wówczas łatwo byłoby obliczyć pierwiastki kwadratowe modulo N poprzez pierwsze rozłożenie na czynniki N , a następnie stosując algorytm omówiony w poprzedniej sekcji. Naszym celem jest teraz pokazanie czegoś odwrotnego: jeśli rozkład na czynniki jest trudny w porównaniu z GenModulus , to samo jest z problemem obliczania pierwiastków kwadratowych. Jeszcze raz podkreślamy, że analogiczny wynik nie jest znany dla problemu RSA ani problemu ustalania rezydualności kwadratowej.

Kluczem jest następujący lemat, który mówi, że dwa „niepowiązane” kwadraty można wykorzystać do pierwiastki dowolnego elementu w Z_N rozłożenia N .

LEMMA 13.35 Niech $N = pq$ z p, q różnymi, nieparzystymi liczbami pierwszymi. Biorąc pod uwagę $x, x^2 \in \mathbb{Z}_N$, możliwe jest rozłożenie N na czynniki w taki sposób, że x wielomianu czasu w N .

DOWÓD Twierdzimy, że albo $\gcd(N, x + x^2) = 1$ albo $\gcd(N, x - x^2) = 1$ jest równe jednemu z czynników pierwszych N .⁶ Ponieważ obliczenia gcd można przeprowadzić w czasie wielomianowym (patrz Załącznik B.1.2), dowodzi to lematu.

⁶ W rzeczywistości oba są równe jednemu z czynników pierwszych N .

Jeśli $x^2 = \hat{x}^2 \pmod{N}$

$$x - \hat{x} \equiv 2 \pmod{N} = (x - \hat{x}) \cdot (x + \hat{x}) \pmod{N},$$

i tak $N \mid (x - \hat{x})(x + \hat{x})$. Następnie $p \mid (x - \hat{x})(x + \hat{x})$ i tak p dzieli jeden z tych wyrazów.

Powiedzmy $p \mid (x + \hat{x})$ (dowód przebiega podobnie, jeśli $p \mid (x - \hat{x})$). Jeśli $q \mid (x + \hat{x})$, to $N \mid (x + \hat{x})$, ale nie może tak być, ponieważ $x = \hat{x} \pmod{N}$.

Zatem $q \mid (x + \hat{x})$ i $\gcd(N, x + \hat{x}) = p$. ■

Alternatywnym sposobem udowodnienia powyższego jest przyjrzenie się temu, co dzieje się w chińskiej reprezentacji pozostałej. Powiedz $x \equiv (xp, xq)$. Następnie, ponieważ x i \hat{x} są pierwiastkami kwadratowymi tej samej wartości y , wiemy, że \hat{x} odpowiada albo $(-xp, xq)$ albo $(xp, -xq)$. (Nie może odpowiadać (xp, xq) ani $(-xp, -xq)$, ponieważ pierwsze odpowiada x , drugie odpowiada $[-x \pmod{N}]$, a obie możliwości są wykluczone przez założenie lematu.) Powiedz $\hat{x} \equiv (-xp, xq)$. Następnie

$$[x + \hat{x} \pmod{N}] = (xp, xq) + (-xp, xq) = (0, [2xq \pmod{q}]),$$

i widzimy, że $x + \hat{x} = 0 \pmod{p}$ podczas gdy $x + \hat{x} = 0 \pmod{q}$. Wynika z tego, że $\gcd(N, x + \hat{x}) = p$, współczynnik N .

Możemy teraz udowodnić główny wynik tej sekcji.

TWIERDZENIE 13.36 Jeżeli rozkład na czynniki jest trudny w porównaniu z GenModulus, to obliczanie pierwiastków kwadratowych jest trudne w porównaniu z GenModulus.

DOWÓD Niech A będzie probabilistycznym algorytmem wielomianowym obliczającym pierwiastki kwadratowe (jak w definicji 13.34). Rozważ następujący probabilistyczny algorytm czasu wielomianowego Afact dla rozkładu modułów wyjściowych przez GenModulus:

Fakt algorytmu :

Algorytm otrzymuje na wejściu moduł N .

- Wybierz mundur $x \in Z_N$ i oblicz $y := [x^2 \pmod{N}]$.
- Uruchom A(N, y), aby uzyskać wynik \hat{x} .
- Jeśli $\hat{x}^2 = y \pmod{N}$ i $\hat{x} = \pm x \pmod{N}$, to należy wziąć pod uwagę N , korzystając z Lematu 13.35.

Z Lematu 13.35 wiemy, że Afact pomyślnie rozkłada N na czynniki dokładnie wtedy, gdy $\hat{x}^2 = \pm x \pmod{N}$ i $\hat{x} = y \pmod{N}$. Oznacza to, że

$$\begin{aligned} \Pr[\text{FactorAfact,GenModulus}(n) = 1] &= \Pr[x^2 = \pm x \pmod{N} \quad \hat{x}^2 = y \pmod{N}] \\ &= \Pr[x^2 = \pm x \pmod{N} \quad x^2 = y \pmod{N} \cdot \Pr[\hat{x}^2 = y \pmod{N}]] = \Pr[\hat{x}^2 = y \pmod{N}], \end{aligned} \quad (13,7)$$

gdzie wszystkie powyższe prawdopodobieństwa odnoszą się do eksperymentu FactorAfact,GenModulus(n) (opis tego eksperymentu można znaleźć w sekcji 8.2.3). W eksperymencie moduł N podany jako dane wejściowe do Afact jest generowany przez GenModulus(1n), a y jest jednolitą resztą kwadratową modulo N, ponieważ x zostało wybrane jednolicie z Z (patrz rozdział 13.4.4). Zatem widok A uruchomiony jako podprogram przez Afact jest rozkładany dokładnie tak samo jak widok A w eksperymencie SQRA,GenModulus(n). Dlatego,

$$\Pr{x^2 \equiv y \pmod{N}} = \Pr{\text{SQRA,GenModulus}(n) = 1}. \quad (13,8)$$

Uwarunkowana wartością reszty kwadratowej y użytej w eksperymencie FactorAfact,GenModulus(n), wartość x z równym prawdopodobieństwem będzie dowolnym z czterech możliwych pierwiastków kwadratowych z y. Oznacza to, że z punktu widzenia algorytmu A (uruchamianego przez Afact jako podprogram) x z równym prawdopodobieństwem będzie każdym z czterech pierwiastków kwadratowych z y. To z kolei oznacza, że pod warunkiem, że A wyprowadzi pierwiastek kwadratowy \hat{x} z y, prawdopodobieństwo, że $\hat{x} = \pm x \pmod{N}$ wynosi (Podkreślamy, że nie przyjmujemy żadnych założeń dotyczących rozkładu \hat{x} wśród pierwiastków kwadratowych z y, a w szczególności nie zakładamy tutaj, że A daje równomierny pierwiastek kwadratowy z y. Używamy raczej faktu, że x ma rozkład jednostajny wśród pierwiastków kwadratowych z y.) Oznacza to, że

$$\Pr{x^2 \equiv y \pmod{N}} = \frac{1}{4}. \quad (13,9)$$

Łącząc równania (13.7)–(13.9) widzimy to

$$\Pr[\text{FactorAfact,GenModulus}(n) = 1] = \frac{1}{4} \cdot \Pr[\text{SQRA,GenModulus}(n) = 1].$$

Ponieważ rozkład na czynnik jest trudny w porównaniu z GenModulus, istnieje pomijalna funkcja negl taka, że

$$\Pr[\text{FactorAfact,GenModulus}(n) = 1] = \text{negl}(n),$$

co implikuje $\Pr[\text{SQRA,GenModulus}(n) = 1] = 2 \cdot \text{negl}(n)$. Ponieważ A było dowolne, kończy to dowód. ■

Poprzednie twierdzenie prowadzi bezpośrednio do rodziny funkcji jednokierunkowych (patrz Definicja 8.76) opartych na dowolnym GenModulusie, względem którego rozkład na czynnik jest trudny:

- Algorystm Gen na wejściu 1n uruchamia GenModulus(1n) w celu otrzymania (N, p, q), a i wyjście I = N. Dziedziną DI jest Z^N zakres RI wynosi QRN.
- Algorystm Samp na wejściu N wybiera element jednolity x ∈ Z^N.
- Algorystm f, na wejściu N i x ∈ Z^N, wyjście [x² mod N].

Powyższe twierdzenie pokazuje, że ta rodzina jest jednokierunkowa, jeśli rozkład na czynnik jest trudny względem GenModulus.

Możemy przekształcić to w rodzinę jednokierunkowych permutacji, używając modułów N specjalnej postaci i pozwalając Dl być podziobrem Z N . (Zobacz ćwiczenie 13.19, aby poznać inny sposób zrobienia z tego permutacji.) Wywołaj N = pq liczbą całkowitą Bluma, jeśli p i q są różnymi liczbami pierwszymi, gdzie p = q = 3 mod 4. Kluczem do zbudowania permutacji jest następujące twierdzenie.

TWIERDZENIE 13.37 Niech N będzie liczbą całkowitą Bluma. Wtedy każda reszta kwadratowa modulo N ma dokładnie jeden pierwiastek kwadratowy, który jest również resztą kwadratową.

DOWÓD Powiedzmy, że N = pq gdzie p = q = 3 mod 4. Korzystając z Twierdzenia 13.17, widzimy, że -1 nie jest resztą kwadratową modulo p lub q. Dzieje się tak, ponieważ dla p = 3 mod 4 utrzymuje się, że p = 4i + 3 dla pewnego i i tak dalej

$$(-1)^{\frac{p-1}{2}} = (-1)^{2i+1} = 1 \text{ mod str}$$

(ponieważ 2i + 1 jest nieparzyste). Niech teraz y (yp, yq) będzie dowolną resztą kwadratową modulo N z czterema pierwiastkami kwadratowymi

$$(xp, xq), (-xp, xq), (xp, -xq), (-xp, -xq).$$

Twierdzimy, że dokładnie jedna z nich jest resztą kwadratową modulo N. Aby to zobaczyć, założymy, że Jp(xp) = +1 i Jq(xq) = -1 (dowód przebiega podobnie w każdym innym przypadku). Stosując Twierdzenie 13.19, mamy Jq(-xq) = Jq(-1) · Jq(xq) =

$$+1 \text{ i tak } (xp, -xq) \text{ odpowiada reszcie ,}$$

kwadratowej modulo N (stosując Twierdzenie 13.21). Podobnie Jp(-xp) = -1, zatem żaden z pozostałych pierwiastków kwadratowych y nie jest resztami kwadratowymi modulo N. ■

Wyrażając to inaczej, powyższa propozycja mówi, że gdy N jest liczbą całkowitą Bluma, to funkcja fN : QRN → QRN dana przez fN(x) = [x mod N] jest permutacją po QRN². Modyfikując algorytm próbkowania Samp powyżej, aby wybrać jednorodny x → QRN (co, jak już widzieliśmy, można zrobić mod N) daje rodzinę łatwo, wybierając jednorodną r → Z ilu jednokierunkowych ponieważ pierwiastki kwadratowe → N i ustawnienie x := [r →]² permutacji. Wreszcie, modulo N można obliczyć w czasie wielomianowym, biorąc pod uwagę faktoryzację N, prosta modyfikacja daje rodzinę permutacji zapadni w oparciu o dowolny moduł GenModulus , w stosunku do którego rozkład na czynniki jest trudny. Nazywa się to czasami rodziną permutacji zapadni Rabina. W podsumowaniu:

TWIERDZENIE 13.38 Niech GenModulus będzie algorytmem, który na wejściu 1 n wyprowadza (N, p, q), gdzie N = pq oraz p i q są różnymi liczbami pierwszymi (z wyjątkiem prawdopodobnie z pomijalnym prawdopodobieństwem) przy p = q = 3 mod 4. Jeśli faktoring jest trudny w porównaniu z GenModulus, wówczas istnieje rodzina permutacji zapadni.

13.5.3 Schemat szyfrowania Rabina

Możemy zastosować wyniki sekcji 13.1.2 do permutacji zapadni Rabina, aby otrzymać schemat szyfrowania klucza publicznego, którego bezpieczeństwo opiera się na faktoring. Aby to zrobić, musimy najpierw zidentyfikować twardy predykat permutacja zapadni. Chociaż moglibyśmy odwołać się do Twierdzenia 13.3, które stwierdza, że zawsze istnieje odpowiedni, twardy predykat, okazuje się, że najmniej znaczący bit lsb jest twardym predykatem dla per-mutacji zapadni Rabina, tak jak ma to miejsce w przypadku RSA (patrz sekcja 11.5.3). Używając tego jako nasz twardy predykat, otrzymujemy schemat konstrukcji 13.39.

BUDOWA 13.39

Niech GenModulus będzie algorytmem wielomianowym, który na wejściu 1n stawia (N, p, q) gdzie $N = pq$ oraz $p \neq q$ są n-bitowymi liczbami pierwszymi (z wyjątkiem prawdopodobieństwa zaniedbywalnego w n) przy $p = q = 3 \pmod{4}$. Skonstruj schemat szyfrowania kluczem publicznym w następujący sposób:

- Gen: na wejściu 1n uruchom GenModulus(1n) aby otrzymać (N, p, q) . The klucz publiczny to N , a klucz prywatny to p, q .
- Enc: po wprowadzeniu klucza publicznego N i wiadomości $m \in \{0, 1\}$, wybierz jednolity $x \in \text{QRN}$ z zastrzeżeniem ograniczenia $\text{lsb}(x) = m$. Wypisz szyfrogram $c := [x^2 \pmod{N}]$.
- Dec: po wprowadzeniu klucza prywatnego p, q i tekstu zaszyfrowanego c , oblicz unikalny $x \in \text{QRN}$ taki, że $x^2 \equiv c \pmod{N}$ i wynik $\text{lsb}(x)$.

Schemat szyfrowania Rabina.

Twierdzenia 13.5 i 13.38 implikują następujący wynik.

TWIERDZENIE 13.40 Jeśli faktoring jest trudny w porównaniu z GenModulus, wówczas Konstrukcja 13.39 jest bezpieczna pod względem CPA.

Szyfrowanie Rabina a szyfrowanie RSA

Warto zwrócić uwagę na podobieństwa i różnice pomiędzy Kryptosystemy Rabina i RSA. (Ta dyskusja dotyczy dowolnej konstrukcji kryptograficznej – niekoniecznie opartej na schemacie szyfrowania klucza publicznego na permutacjach zapadni Rabina lub RSA.)

Permutacje zapadni RSA i Rabina wydają się dość podobne, z podniesieniem do kwadratu w przypadku Rabina odpowiadające przyjęciu $e = 2$ w tym przypadku z RSA. (Oczywiście 2 nie jest liczbą pierwszą względnie pierwszą względem $\phi(N)$, więc Rabin nie jest szczególny przypadek RSA.) Pod względem bezpieczeństwa oferowanego przez każdą konstrukcję, twardość obliczania modułowych pierwiastków kwadratowych jest równoważna twardości faktoringu, natomiast nie jest znana trudność rozwiązania problemu RSA

wynikać z trudności faktoringu. Permutacja zapadni Rabina opiera się zatem na potencjalnie słabszym założeniu: teoretycznie jest możliwe, że ktoś opracuje skuteczny algorytm rozwiązania problemu RSA, ale obliczenie pierwiastków kwadratowych pozostanie trudne. Lub ktoś może opracować algorytm, który rozwiąże problem RSA szybciej niż znane algorytmy faktoringu.

Lemat 13.35 zapewnia jednak, że obliczanie pierwiastków kwadratowych modulo N nigdy nie będzie dużo szybsze niż najlepszy dostępny algorytm rozkładu na czynniki N.

Pod względem wydajności permutacje RSA i Rabina są zasadniczo takie same. Właściwie, jeśli w przypadku RSA zostanie użyty duży wykładnik e, wówczas obliczenie potęg e^m (jak w RSA) jest nieco wolniejsze niż podnoszenie do kwadratu (jak u Rabina). Z drugiej strony, wymagana jest nieco większa ostrożność podczas pracy z permutacją Rabina, ponieważ jest to tylko permutacja na podzbiorze Z w przeciwieństwie do RSA, co daje permutację na całym Z .

Schemat szyfrowania „zwykłego Rabina”, skonstruowany w sposób analogiczny do zwykłego szyfrowania RSA, jest podatny na atak wybranym szyfrogramem, który umożliwia przeciwnikowi poznanie całego klucza prywatnego (patrz ćwiczenie 13.17). Choć zwykły RSA również nie jest zabezpieczony CCA, ataki ze znany wybranym tekstem zaszyfrowanym na zwykły RSA są mniej szkodliwe, ponieważ pozwalają odzyskać wiadomość, ale nie klucz prywatny. Być może istnienie takiego ataku na „zwykłego Rabina” wpłynęło na kryptografów już na początku, aby całkowicie odrzucić użycie szyfrowania Rabina.

Podsumowując, permutacja RSA jest znacznie szerzej stosowana w praktyce niż permutacja Rabina, ale w świetle powyższego wydaje się, że wynika to bardziej z przypadku historycznego niż z jakiegokolwiek przekonującego uzasadnienia technicznego.

Referencje i dodatkowe lektury

Istnienie szyfrowania klucza publicznego opartego na dowolnych permutacjach zapadni wykazał Yao [179], a poprawę wydajności omówioną na końcu sekcji 13.1.2 zawdzięczają Blumowi i Goldwasserowi [36].

Childs [44] i Shoup [159] dostarczają dalszych informacji na temat (obliczeniowej) teorii liczb użytej w tym rozdziale. Dobry opis algorytmu obliczania symbolu Jacobiego modulo złożonego o nieznanej faktoryzacji wraz z dowodem poprawności podano w [57].

Schemat szyfrowania Pailliera został wprowadzony w [136]. Shoup [159, Sec-dla 7.5] daje charakterystykę Z po prostu $N = pq_{\text{nie}}$ dowolnych liczb całkowitych N , e (i nie $e = 2$, jak zrobiono tutaj).

Problem wyznaczania resztości kwadratowej modulo złożonej z nieznanej faktoryzacji sięga Gaussa [71] i jest powiązany z innymi (domniemanymi) twardymi problemami z zakresu teorii liczb. Schemat szyfrowania Goldwassera-Micali [80], wprowadzony w 1982 r., był pierwszym schematem szyfrowania kluczem publicznym z dowodem bezpieczeństwa.

Rabin [145] wykazał, że obliczanie pierwiastków kwadratowych modulo kompozytu jest równoznaczne z faktoringiem. Wyniki rozdziału 13.5.2 pochodzą od Bluma [35]. Twarde predykaty dla permutacji zapadni Rabina omówiono w [8, 86, 7] i zawartych tam źródłach.

Ćwiczenia

13.1 Skonstruuj i udowodnij zabezpieczenie CPA dla KEM w oparciu o dowolną permutację zapadni poprzez odpowiednie uogólnienie konstrukcji 11.34.

13.2 Pokaż, że izomorfizm Twierdzenia 13.6 można skutecznie odwrócić, gdy znana jest faktoryzacja N .

13.3 Niech $\Phi(N)$ oznacza zbiór $\{(a, 1) \mid a \in Z_N\}$. Z trudno określić, czy dany element $y \in Z_N$ jest resztą kwadratową w grupie dodatków Z_N .

13.4 Niech G będzie grupą abelową. Pokaż, że zbiór reszt kwadratowych w G tworzy podgrupę.

13.5 To pytanie dotyczy reszt kwadratowych w grupie dodatków Z_N .

(Element $y \in Z_N$ jest resztą kwadratową wtedy i tylko wtedy, gdy istnieje $x \in Z_N$ z $2x = y \pmod{N}$.)

(a) Niech p będzie liczbą pierwszą nieparzystą. Ile elementów Z_p jest kwadratowych pozostałości?

(b) Niech $N = pq$ będzie iloczynem dwóch nieparzystych liczb pierwszych p i q . Ile elementy Z_N są resztami kwadratowymi?

(c) Niech N będzie parzystą liczbą całkowitą. Ile elementów Z_N jest kwadratowych pozostałości?

13.6 Niech $N = pq$ z p, q różnymi, nieparzystymi liczbami pierwszymi. Pokaż algorytm ppt wybierania jednolitego elementu $QN R+1$, gdy znana jest faktoryzacja N . (Twój algorytm może mieć znikome prawdopodobieństwo awarii w N .)

13.7 Niech $N = pq$ z p, q różnymi, nieparzystymi liczbami pierwszymi. Udowodnić, że jeśli $[x^{-1} R+1 \pmod{N}] = QRN$, i jeśli $x \in QN R+1$ to $[x^{-1} \pmod{N}] = QRN$.

13.8 Niech $N = pq$ z p, q różnymi, nieparzystymi liczbami pierwszymi i ustalimy $z \in QN R+1 \cap N$. Pokaż, że wybierając uniform $x \in QRN$ i ustawnienie $y := [z \cdot x \pmod{N}]$ otrzymujemy $ay \in$ rozkładzie jednostajnym w $QN R+1 \cap N$. Oznacza to, że dla dowolnego $\hat{y} \in QN R+1 \cap N$

$$\Pr[z \cdot x = \hat{y} \pmod{N}] = 1 / |QN R+1 \cap N|,$$

gdzie przyjmuje się prawdopodobieństwo jednolitego wyboru $x \in QRN$.

Wskazówka: skorzystaj z poprzedniego ćwiczenia.

- 13.9 Niech N będzie iloczynem 5 różnych, nieparzystych liczb pierwszych. Jeśli $y \equiv z_N^2 \pmod{N}$, to kwadrat $= y$ pozostałość, ile rozwiązań ma równanie $x^2 \pmod{N}$?
- 13.10 Pokaż, że schemat szyfrowania Goldwassera-Micaliego jest homomorficzny, jeśli przestrzeń wiadomości $\{0, 1\}$ postrzegamy jako grupę Z_2 .
- 13.11 Rozważmy następującą odmianę schematu szyfrowania Goldwassera-Micali: GenModulus($1n$) jest uruchamiany w celu uzyskania (N, p, q) , gdzie $N = pq$ i $p = q = 3 \pmod{4}$. (Tj. N jest liczbą całkowitą Bluma.) Klucz publiczny to N , a klucz prywatny to p, q . Aby zaszyfrować $m \in \{0, 1\}$, nadawca wybiera uniform $x \in Z_N$ i oblicza szyfrogram $c := [(-1)^m \cdot x \pmod{N}]$.
 (a) Udowodnij, że dla N podanej postaci, $[-1 \pmod{N}] = QN R+1$ (b) Udowodnij, że opisany schemat ma nieroróżnialne szyfrowanie w przypadku ataku wybranym tekstem jawnym, jeśli określenie resztości kwadratowej jest trudne w porównaniu z GenModulus.
- 13.12 Założmy, że określenie resztości kwadratowej jest trudne dla GenModulus. Pokaż, że implikuje to trudność odróżnienia jednolitego elementu $+1 \pmod{N}$ od jednolitego elementu \underbrace{N}_{N} .
- 13.13 Pokaż, że zwykłe szyfrowanie RSA wiadomości m powoduje wyciek $JN(m)$.
- 13.14 Rozważmy następującą odmianę schematu szyfrowania Goldwassera-Micali: GenModulus($1n$) jest uruchamiany w celu uzyskania (N, p, q) . Klucz publiczny to N , a klucz prywatny to p, q . Aby zaszyfrować 0, nadawca wybiera n jednolitych elementów $c_1, \dots, c_n \in QRN$. Aby zaszyfrować 1, nadawca wybiera n jednolitych elementów $c_1, \dots, c_n \in Jot N$. W każdym przypadku wynikowy tekst zaszyfrowany to $c = c_1, \dots, c_n$.
 (a) Pokaż, jak nadawca może wygenerować jednolity element czasu $\underbrace{N}^{+1} W$ wielomianowego J , w przypadku którego prawdopodobieństwo niepowodzenia jest znikome. (b) Zaproponuj odbiorca sposób skutecznego odszyfrowania, choć z znikomym prawdopodobieństwem błędu. (c) Udowodnić, że jeśli określenie resztości kwadratowej jest trudne w porównaniu z GenModulus, ten schemat jest bezpieczny w CPA.
 Wskazówka: skorzystaj z poprzedniego ćwiczenia.
- 13.15 Niech G będzie algorytmem czasu wielomianowego, który na wejściu $1n$ daje na wyjściu liczbę p z $p = n$ generator g Z nie jest trudny w_p pierwszą. Udowodnić, że problem DDH stosunku do G .
 Wskazówka: Skorzystaj z faktu, że rezydualność kwadratową można skutecznie określić modulo liczby pierwszej.
- 13.16 Uważa się, że problem logarytmu dyskretnego jest trudny dla G , podobnie jak w poprzednim ćwiczeniu. Oznacza to, że funkcja (rodzina) $f_{p,g}$ gdzie $f_{p,g}(x)$ bit x . Pokaż, że lsb nie jest twardym $\stackrel{\text{def}}{=} [g \cdot x \pmod{p}]$ jest jednokierunkowe. Niech $lsb(x)$ oznacza najmniej znaczącą predykatem dla $f_{p,g}$.

- 13.17 Rozważmy prosty schemat szyfrowania Rabina, w którym wiadomość m QRN jest szyfrowany względem klucza publicznego N (gdzie N jest liczbą całkowitą Bluma) poprzez obliczenie tekstu zaszyfrowanego $c := [m^2 \bmod N]$. Pokaż atak wybranym szyfrogramem na ten schemat, który odzyskuje cały klucz prywatny.
- 13.18 Zwykły schemat podpisu Rabina jest podobny do zwykłego schematu podpisu RSA, z wyjątkiem użycia permutacji zapadni Rabina. Pokaż atak na równinie Podpisy Rabina, dzięki którym atakujący poznaje klucz prywatny osoby podpisującej.
- 13.19 Niech N będzie liczbą całkowitą Bluma.

(a) Zdefiniuj zbiór $S = \{x \in \mathbb{Z}_N \mid x < N/2 \text{ i } JN(x) = +1\}$. Definiować funkcję $fN : S \rightarrow \mathbb{Z}_N$ przez:

$$fN(x) = \begin{cases} [x^2 \bmod N] \text{ jeśli } [x^2 \bmod N] < N/2 \\ [-x^2 \bmod N] \text{ jeśli } [x^2 \bmod N] > N/2 \end{cases}$$

Pokaż, że fN jest permutacją po S .

(b) Zdefiniuj rodzinę permutacji zapadni w oparciu o użycie faktoringu fN jak zdefiniowano powyżej.

- 13.20 Niech N będzie liczbą całkowitą Bluma. Zdefiniuj funkcję $halfN : \mathbb{Z}_N \rightarrow \{0, 1\}$ jak

$$\text{połowaN}(x) = \begin{cases} 1 \text{ jeśli } x < N/2 \\ +1 \text{ jeśli } x > N/2 \end{cases}$$

Pokaż, że funkcja $f : \mathbb{Z}_N \rightarrow QRN \times \{-1, +1\}^2$ zdefiniowana jako

$$f(x) = [x^2 \bmod N], JN(x), \text{półN}(x)$$

jest jeden do jednego.

Machine Translated by Google

Indeks wspólnej notacji

Notacja ogólna:

- \coloneqq odnosi się do przypisania deterministycznego
- Jeśli S jest zbiorem, to $x \in S$ oznacza, że x jest wybrane jednostajnie z S
- Jeśli A jest algorytmem losowym, to $y = A(x)$ oznacza uruchomienie A na wejściu x jednolitą losową taśmą i przypisanie wyjścia do y . Piszemy $y := A(x; r)$, aby oznaczyć uruchomienie A na wejściu x przy użyciu losowej taśmy r i przypisanie wyjścia do y
- \wedge oznacza koniunkcję logiczną (operator AND)
- \vee oznacza alternatywę boolowską (operator OR)
- \oplus oznacza operator wyłączności lub (XOR); operator ten można zastosować do pojedynczych bitów lub całych ciągów znaków (w tym drugim przypadku XOR jest bitowy)
- $\{0, 1\}^n$ jest zbiorem wszystkich ciągów bitów o długości n
- $\{0, 1\} \leq n$ jest zbiorem wszystkich ciągów bitów o długości co najwyżej n
- $\{0, 1\}^*$ to zbiór wszystkich skończonych ciągów bitów
 - n (odp. 1^n) oznacza ciąg składający się z n zer (odp. n jedynek) • 0^n
- x oznacza długość binarnej reprezentacji (dodatniej) liczby całkowitej x , zapisanej z bitem wiodącym 1. Należy zauważyć, że $\log x < x \leq \log x + 1$
- $|x|$ oznacza długość ciągu binarnego x (który może mieć początkowe zera), lub wartość bezwzględna liczby rzeczywistej x
- $O(\cdot)$, $\Theta(\cdot)$, $\Omega(\cdot)$, $\mathcal{W}(\cdot)$ patrz Załącznik A.2
- $0x$ oznacza, że cyfry są reprezentowane w formacie szesnastkowym
- xy oznacza jednoznaczne połączenie ciągów x i y („niejednoznaczne” oznacza, że x i y można odzyskać z xy)
- $\Pr[X]$ oznacza prawdopodobieństwo zdarzenia X
- $\log x$ oznacza logarytm o podstawie 2 x

Notacja specyficzna dla kryptowalut:

- n jest parametrem bezpieczeństwa
- ppt oznacza „probabilistyczny czas wielomianowy”
- $\text{AO}(\cdot)$ oznacza algorytm A z dostępem Oracle do O
- k zazwyczaj oznacza tajny klucz (jak w przypadku szyfrowania kluczem prywatnym i adresów MAC)
- (pk, sk) oznacza parę kluczy publiczny/prywatny (do szyfrowania kluczem publicznym i Podpisy cyfrowe)
- \negl oznacza funkcję pomijalną; patrz Definicja 3.4
- $\text{poli}(n)$ oznacza dowolny wielomian
- $\text{polilog}(n)$ oznacza $\text{poli}(\log(n))$
- Funcn oznacza zbiór funkcji odwzorowujących ciągi n -bitowe na ciągi n -bitowe
- Permn oznacza zbiór bijekcji na n -bitowych ciągach
- IV oznacza wektor inicjujący (używany dla trybów działania i funkcji skrótu odpornych na kolizje)

Algorytmy i procedury:

- G oznacza generator pseudolosowy
- F oznacza funkcję z kluczem, która jest zazwyczaj funkcją pseudolosową lub permutacją
- $(\text{Gen}, \text{Enc}, \text{Dec})$ oznaczają odpowiednio procedury generowania, szyfrowania i deszyfrowania klucza, zarówno dla szyfrowania klucza prywatnego, jak i publicznego.
W przypadku szyfrowania kluczem prywatnym, gdy Gen nie jest określony, wówczas $\text{Gen}(1n)$ generuje jednolite $k \in \{0, 1\}^n$
- $(\text{Gen}, \text{Mac}, \text{Vrfy})$ oznaczają odpowiednio procedury generowania klucza, generowania tagu i weryfikacji dla kodu uwierzytelniającego wiadomość. Gdy Gen jest nieokreślony, wówczas $\text{Gen}(1n)$ generuje jednolite $k \in \{0, 1\}^n$
- $(\text{Gen}, \text{Sign}, \text{Vrfy})$ oznaczają odpowiednio procedury generowania klucza, generowania podpisu i weryfikacji dla schematu podpisu cyfrowego
- GenPrime oznacza algorytm ppt , który na wejściu $1n$ generuje n -bitową liczbę pierwszą, z wyjątkiem przypadku, gdy prawdopodobieństwo jest znikome w n
- GenModulus oznacza algorytm ppt , który na wejściu $1n$ wyprowadza (N, p, q) , gdzie $N = pq$ i (z zaniedbywalnym prawdopodobieństwem) p i q są n -bitowymi liczbami pierwszymi

- GenRSA oznacza algorytm ppt , który na wejściu 1n wyprowadza (z wyjątkiem znikomego prawdopodobieństwa) moduł N, liczbę całkowitą $e > 0$ z $\gcd(e, \varphi(N)) = 1$ i liczbę całkowitą d spełniającą $ed = 1 \pmod{\varphi(N)}$
- G oznacza algorytm ppt , który na wejściu 1n wyprowadza (z wyjątkiem znikomego prawdopodobieństwa) opis grupy cyklicznej G, rzad grupowy q (przy $q = n$) i generator g G.

Teoria liczb:

- Z oznacza zbiór liczb całkowitych
- $| b$ oznacza a dzieli b
- $| b$ oznacza, że a nie dzieli b
- $\gcd(a, b)$ oznacza największy wspólny dzielnik aib
- $[a \pmod b]$ oznacza resztę a z dzielenia przez b. Zauważ to
 $0 \leq [a \pmod b] < b$.
- $x_1 = x_2 = \dots = x_n \pmod N$ oznacza, że x_1, \dots, x_n są przystające modulo N

Uwaga: $x = y \pmod N$ oznacza, że x i y są przystające modulo N, podczas gdy $x = [y \pmod N]$ oznacza, że x równa się reszcie z y podzielonej przez N.

- Z_N oznacza grupę addytywną liczb całkowitych modulo N oraz zbiór $\{0, \dots, N - 1\}$
- Z oznacza multiplikatywną grupę odwracalnych liczb całkowitych modulo N N (tzn. takich, które są względnie pierwsze do N)
- $\varphi(N)$ oznacza rozmiar Z_{N}
- G i H oznaczają grupy
- $G_1 \cong G_2$ oznacza, że grupy G_1 i G_2 są izomorficzne. Jeśli ten izomorfizm jest dany przez $f : G_1 \rightarrow G_2$, to piszemy $x_1 \cong x_2$
- g jest zazwyczaj generatorem grupy
- $\log_g h$ oznacza logarytm dyskretny h przy podstawie g
- g oznacza grupę utworzoną przez g
- p i q zwykle oznaczają liczby pierwsze
- N zazwyczaj oznacza iloczyn dwóch różnych liczb pierwszych p i q równych długość

- QR_p jest zbiorem reszt kwadratowych modulo p
- QN_{Rp} jest zbiorem niereszt kwadratowych modulo p
- $J_p(x)$ jest symbolem Jacobiego x modulo p
- J_N^{+1} jest zbiorem elementów o symbolu Jacobiego +1 modulo N
- J_N^{-1} jest zbiorem elementów o symbolu Jacobiego -1 modulo N
- QN_{R+1_N} jest zbiorem kwadratowych niereszt modulo N mających Jacobiego symbol +1

załącznik A

Tło matematyczne

A.1 Tożsamości i nierówności

Podajemy kilka standardowych tożsamości i nierówności, które są używane w różnych miejscach tekstu.

TWIERDZENIE A.1 (Twierdzenie o rozwinięciu dwumianowym) Niech x, y będą liczbami rzeczywistymi i niech n będzie dodatnią liczbą całkowitą. Następnie

$$(x + y)^n = \sum_{j=0}^n \binom{n}{j} x^{n-j} y^j.$$

TWIERDZENIE A.2 Dla każdego $x > 1$ utrzymuje się, że $(1 - 1/x)^{1-x} < e^{-1}$.

TWIERDZENIE A.3 Dla każdego x utrzymuje się, że $1 - x \leq e^{-x}$.

TWIERDZENIE A.4 Zachodzi to dla wszystkich $x \in (0, 1)$

$$\frac{\ln(1-x)}{x} \geq -1 - \frac{x}{1-x}.$$

A.2 Notacja asymptotyczna

Do wyrażania asymptotycznego zachowania funkcji używamy standardowej notacji.

DEFINICJA A.5 Niech $f(n), g(n)$ będą funkcjami od nieujemnych liczb całkowitych do nieujemnych liczb rzeczywistych. Następnie:

- $f(n) = O(g(n))$ oznacza, że istnieją dodatnie liczby całkowite c i n takie, że dla wszystkich $n > n_0$ zachodzi zasada, że $f(n) \leq c \cdot g(n)$.

- $f(n) = \Omega(g(n))$ oznacza, że istnieją dodatnie liczby całkowite c i n takie, że dla wszystkich $n > n$ zachodzi zasada, że $f(n) \geq c \cdot g(n)$.
- $f(n) = \Theta(g(n))$ oznacza, że istnieją dodatnie liczby całkowite c_1, c_2 i n takie, że dla wszystkich $n > n$ zachodzi zasada $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$.
- $f(n) = o(g(n))$ oznacza, że $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$.
- $f(n) = \omega(g(n))$ oznacza, że $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$.

Przykład A.6

Niech $f(n) = n^4 + 3n + 500$. Następnie:

- $f(n) = O(n^4)$.
- $f(n) = O(n^5)$. W rzeczywistości $f(n) = o(n^5)$.
- $f(n) = \Omega(n^3 \log n)$. W rzeczywistości $f(n) = \omega(n^3 \log n)$.
- $f(n) = \Theta(n^4)$.

A.3 Podstawowe prawdopodobieństwo

Zakładamy, że czytelnik jest zaznajomiony z podstawową teorią prawdopodobieństwa na poziomie tego, co jest omawiane na typowym kursie licencjackim z matematyki dyskretnej. Tutaj po prostu przypominamy czytelnikowi pewne oznaczenia i podstawowe fakty.

Jeżeli E jest zdarzeniem, to E^- oznacza dopełnienie tego zdarzenia; tj. E^- jest zdarzeniem, w którym E nie występuje. Z definicji $\Pr[E] = 1 - \Pr[E^-]$. Jeżeli E_1 i E_2 są zdarzeniami, to $E_1 \cap E_2$ oznacza ich koniunkcję; tj. $E_1 \cap E_2$ jest zdarzeniem, w którym występują zarówno E_1 , jak i E_2 . Z definicji $\Pr[E_1 \cap E_2] = \Pr[E_1] \cdot \Pr[E_2]$. Mówiąc, że zdarzenia E_1 i E_2 są niezależne, jeśli $\Pr[E_1 \cap E_2] = \Pr[E_1] \cdot \Pr[E_2]$.

Jeżeli E_1 i E_2 są zdarzeniami, to $E_1 \cup E_2$ oznacza alternatywę E_1 i E_2 ; oznacza to, że $E_1 \cup E_2$ jest zdarzeniem, w którym występuje E_1 lub E_2 . Z definicji wynika, że $\Pr[E_1 \cup E_2] = \Pr[E_1] + \Pr[E_2]$. Granica unii jest często bardzo przydatną górną granicą tej wielkości.

PROPOZYCJA A.7 (związany z Unią)

$$\Pr[E_1 \cup E_2] \leq \Pr[E_1] + \Pr[E_2].$$

Wielokrotne zastosowanie związku związanego dla dowolnych zdarzeń E_1, \dots, E_k daje

$$\Pr \left[\bigcap_{j=1}^k E_j \right] = \prod_{j=1}^k \Pr[E_j].$$

Prawdopodobieństwo warunkowe E_1 przy danym E_2 , oznaczone jako $\Pr[E_1 | E_2]$, definiuje się jako

$$\Pr[E_1 | E_2] \stackrel{\text{def}}{=} \frac{\Pr[E_1 \cap E_2]}{\Pr[E_2]}$$

o ile $\Pr[E_2] = 0$. (Jeśli $\Pr[E_2] = 0$, to $\Pr[E_1 | E_2]$ jest niezdefiniowane.) Reprezentuje to prawdopodobieństwo wystąpienia zdarzenia E_1 , przy założeniu, że wystąpiło zdarzenie E_2 . Z definicji wynika od razu, że

$$\Pr[E_1 \cap E_2] = \Pr[E_1 | E_2] \cdot \Pr[E_2];$$

równość zachodzi nawet wtedy, gdy $\Pr[E_2] = 0$, o ile w oczywisty sposób interpretujemy mnożenie przez zero po prawej stronie.

Mogliśmy teraz łatwo wyrowadzić twierdzenie Bayesa.

TWIERDZENIE A.8 (Twierdzenie Bayesa) Jeżeli $\Pr[E_2] = 0$ to

$$\Pr[E_1 | E_2] = \frac{\Pr[E_2 | E_1] \cdot \Pr[E_1]}{\Pr[E_2]}.$$

DOWÓD Wynika z tego, ponieważ

$$\Pr[E_1 | E_2] = \frac{\Pr[E_1 \cap E_2]}{\Pr[E_2]} = \frac{\Pr[E_2 | E_1] \cdot \Pr[E_1]}{\Pr[E_2]} = \frac{\Pr[E_2 | E_1] \cdot \Pr[E_1]}{\Pr[E_2]}.$$



Niech E_1, \dots, E_n będą zdarzeniami takimi, że $\Pr[E_1 \cap \dots \cap E_n] = 1$ i $\Pr[E_i \cap E_j] = 0$ dla wszystkich $i \neq j$. Oznacza to, że $\{E_i\}$ dzieli przestrzeń wszystkich możliwych zdarzeń w taki sposób, że z prawdopodobieństwem 1 zajdzie dokładnie jedno ze zdarzeń E_i . Następnie dla dowolnego F

$$\Pr[F] = \prod_{i=1}^n \Pr[F \cap E_i].$$

Szczególnym przypadkiem jest sytuacja, gdy $n = 2$ i $E_2 = E^-$, co daje

$$\begin{aligned} \Pr[F] &= \Pr[F \cap E_1] + \Pr[F \cap E^-] \\ &= \Pr[F | E_1] \cdot \Pr[E_1] + \Pr[F | E^-] \cdot \Pr[E^-]. \end{aligned}$$

Biorąc $F = E_1 \cup E_2$, otrzymujemy ścisłejszą wersję związku:

$$\begin{aligned} \Pr[E_1 \cup E_2] &= \Pr[E_1 \cap E_2 | E_1] \cdot \Pr[E_1] + \Pr[E_1 \cap E_2 | E^-] \cdot \Pr[E^-] + \Pr[E_2 | E^-] \\ &+ \Pr[E_2 | E^-]. \end{aligned}$$

Rozszerzając to na zdarzenia E_1, \dots, E_n otrzymujemy

PROPOZYCJA A.9

$$\Pr_{\substack{i=1 \\ j=2}} \left(E_i \cap E_j \right) = \Pr[E_1] + \Pr[E_2] + \dots + \Pr[E_i | E_1 \cap E_2 \cap \dots \cap E_{i-1}]$$

* Przydatne granice prawdopodobieństwa

Przeglądamy pewną terminologię i granice prawdopodobieństwa, które są standardowe, ale mogą nie zostać napotkane na podstawowym kursie matematyki dyskretnej.

Materiał ten został użyty jedynie w Rozdziale 7.3.

(dyskretna, o wartościach rzeczywistych) zmienna losowa X to zmienna, której wartość jest przypisana probabilistycznie na podstawie pewnego skończonego zbioru S liczb rzeczywistych. X jest nieujemne, jeśli nie przyjmuje wartości ujemnych; jest to zmienna losowa $0/1$, jeśli $S = \{0, 1\}$. Zmienne losowe $0/1 X_1, \dots, X_k$ są niezależne, jeżeli dla b_1, \dots, b_k zachodzi, że $\Pr[X_1 = b_1 \cap \dots \cap X_k = b_k] = \text{wszystkie } \Pr[X_i = b_i]$.

Niech $\text{Exp}[X]$ oznacza oczekiwanie zmiennej losowej X ; jeśli X przyjmuje wartości ze zbioru S , to $\text{Exp}[X] = \sum_{s \in S} s \cdot \Pr[X = s]$. Jednym z najważniejszych faktów jest to, że oczekiwania są liniowe; dla zmiennych losowych X_1, \dots, X_k (z dowolnymi zależnościami) mamy $\text{Exp}[X_1 + \dots + X_k] = \text{Exp}[X_1] + \dots + \text{Exp}[X_k]$. Jeżeli X_1, X_2 są niezależne, to $\text{Exp}[X_1 \cdot X_2] = \text{Exp}[X_1] \cdot \text{Exp}[X_2]$

Nierówność Markowa jest przydatna, gdy niewiele wiadomo o X .

TWIERDZENIE A.10 (nierówność Markowa) Niech X będzie nieujemną zmienną losową i $v > 0$. Wtedy $\Pr[X \geq v] \leq \text{Exp}[X]/v$.

DOWÓD Założmy, że X przyjmuje wartości ze zbioru S . Mamy

$$\begin{aligned} \text{Eksp}[X] &= \sum_{s \in S} s \cdot \Pr[X = s] \\ &= \sum_{x \in S, x \leq v} \Pr[X = s] \cdot 0 + \sum_{x \in S, x > v} v \cdot \Pr[X = s] \\ &= v \cdot \Pr[X \geq v]. \end{aligned}$$



Wariancja X , oznaczona jako $\text{Var}[X]$, mierzy, jak bardzo X odchodzi od jego oczekiwania. Mamy $\text{Var}[X] = \text{Exp}[(X - \text{Exp}[X])^2] = \text{Exp}[X^2] - \text{Exp}[X]^2$, można łatwo pokazać, że $\text{Var}[aX + b] = a^2 \text{Var}[X]$. Dla losowości $0/1$ mamy $\text{Var}[X_i] = 1/4$, $\text{Exp}[X_i](1 - \text{Exp}[X_i])$ (także jest to zgodne z $\text{Exp}[X_i]$, gdyż $\text{Exp}[X_i]$ jest zmienna X_i) i tak $\text{Var}[X_i] =$

TWIERDZENIE A.11 (nierówność Czebyszewa) Niech X będzie zmienną losową, a $\delta > 0$. Wtedy:

$$\Pr[|X - \text{Exp}[X]| \geq \delta] \leq \frac{\text{Var}[X]}{\delta^2}.$$

DOWÓD Zdefiniuj nieujemną zmienną losową Y , a następnie $Y = (X - \text{Exp}[X])^2$ i zastosuj nierówność Markowa. Więc,

$$\Pr[|X - \text{Exp}[X]| \geq \delta] = \Pr[(X - \text{Exp}[X])^2 \geq \delta^2] = \Pr[2 \frac{\text{Exp}[X] - \text{Exp}[X]^2}{2} \geq \delta^2] = \frac{\text{Var}[X]}{\delta^2}.$$



Zmienne losowe $0/1 X_1, \dots, X_m$ są paraminiezależne, jeśli dla każdego $i, j \in \{0, 1\}$ zachodzi, że $i = j$ i każde bi

$$\Pr[X_i = b_i \cap X_j = b_j] = \Pr[X_i = b_i] \cdot \Pr[X_j = b_j].$$

Jeśli X_1, \dots, X_m są parami niezależne, to $\text{Var}[X_i] = \text{Var}[X_j]$.

(Wynika to z faktu, że $\text{Exp}[X_i \cdot X_j] = \text{Exp}[X_i] \cdot \text{Exp}[X_j]$, gdy $i = j$, stosując niezależność parami.) Następuje ważny wniosek z nierówności Czebyszewa.

DODATEK A.12 Niech X_1, \dots, X_m będą parami niezależnymi zmiennymi losowymi o takich samych oczekiwaniach μ i wariancji σ^2 . Następnie dla każdego $\delta > 0$,

$$\Pr\left[\left|\frac{\sum_{j=1}^M X_j}{M} - \mu\right| \geq \delta\right] \leq \frac{2\sigma^2}{\delta^2}.$$

DOWÓD Z liniowością oczekiwania, $\text{Exp}\left[\sum_{j=1}^M X_j/m\right] = \mu$. Stosując X_i/m , mamy

$$\Pr\left[\left|\frac{\sum_{j=1}^M X_j}{M} - \mu\right| \geq \delta\right] \leq \frac{\text{Var}\left[\frac{1}{M} \sum_{j=1}^M X_j\right]}{\delta^2}.$$

Wynika z tego, korzystając z niezależności par

$$\text{Var}\left[\frac{1}{M} \sum_{j=1}^M X_j\right] = \frac{1}{M} \sum_{j=1}^M \text{Var}[X_j] = \frac{1}{M} \sum_{j=1}^M \frac{1}{m^2} \sum_{i=1}^m \Pr[X_{ij} \neq m] = \frac{2\sigma^2}{M}.$$

Nierówność uzyskuje się poprzez połączenie dwóch powyższych równań.



Powiedzmy, że zmienne losowe $0/1 X_1, \dots, X_m$ każdy zapewnia oszacowanie pewnego ustalonego (nieznanego) bitu b . Oznacza to, że $\Pr[X_i = b] = 1/2 + \epsilon$ dla wszystkich i , gdzie $\epsilon > 0$.

Możemy oszacować b , patrząc na wartość X_1 ; oszacowanie to będzie prawidłowe z prawdopodobieństwem $\Pr[X_1 = b]$. Lepsze oszacowanie można uzyskać, patrząc na wartości X_1, \dots, X_m i przyjmując wartość, która występuje najczęściej. Analizujemy, jak dobrze to działa, gdy X_1, \dots, X_m są parami niezależne.

TWIERDZENIE A.13 Ustal $\epsilon > 0$ i $b \in \{0, 1\}$ i niech $\{X_i\}$ będzie parami niezależnymi zmiennymi losowymi $0/1$, dla których $\Pr[X_i = b] = +\epsilon$ dla wszystkich $i \in \overline{1, n}$. Rozważmy proces, w którym m wartości X_1, \dots, X_m są rejestrowane, a X jest ustawiane na wartość, która występuje w zdecydowanej większości przypadków. Następnie

$$\Pr[X = b] \geq \frac{1}{24 \cdot \epsilon \cdot m}.$$

DOWÓD Założymy, że $b = 1$; dzięki symetrii dzieje się to bez utraty ogólności.

Wtedy $\Pr[X_i = 1] = +\epsilon$. Niech X oznacza zdecydowaną większość $\{X_i\}$ jak w twierdzeniu i zauważ, że $X = 1$ wtedy i tylko wtedy, gdy $X_i = 1$ dla $i = 1, \dots, m/2$. Więc

$$\begin{aligned} \Pr[X = 1] &= \Pr \left[\bigvee_{i=1}^{m/2} X_i = 1 \right] \\ &= \Pr \left[\frac{\sum_{i=1}^{m/2} X_i}{m/2} \geq \frac{1}{2} \right] \\ &= \Pr \left[\frac{\sum_{i=1}^{m/2} X_i}{m} \geq \frac{1}{2} + \epsilon \right] \\ &\geq \Pr \left[\frac{\sum_{i=1}^m X_i}{m} \geq \frac{1}{2} + \epsilon \right]. \end{aligned}$$

Ponieważ $\text{Var}[X_i] = 1/4$ dla wszystkich i , zastosowanie poprzedniego wniosku pokazuje, że $\Pr[X = 1] \geq \frac{1}{24 \cdot \epsilon \cdot m}$.

Lepszą granicę uzyskuje się, jeśli $\{X_i\}$ są niezależne:

TWIERDZENIE A.14 (ograniczony Chernoffa) Ustal $\epsilon > 0$ i $b \in \{0, 1\}$ i niech $\{X_i\}$ będzie niezależnymi zmiennymi losowymi $0/1$ z $\Pr[X_i = b] = +\epsilon$ dla wszystkich $i \in \overline{1, n}$. Prawdopodobieństwo, że ich wartość większościowa nie wynosi b , wynosi co najwyżej

A.4 Problem „urodzin” Jeśli wybierzemy q

elementów y_1, \dots, y_q równomiernie ze zbioru o rozmiarze N , jakie jest prawdopodobieństwo, że istnieją różne i, j z $y_i = y_j$? Odnosimy się do podanych

zdarzenie jako kolizję i oznacza prawdopodobieństwo tego zdarzenia przez $\text{col}(q, N)$. Problem ten jest powiązany z tak zwanym problemem urodzinowym, który pyta, jakiej wielkości grupy osób potrzebujemy, aby z prawdopodobieństwem $1/2$ jakaś para osób w grupie obchodziła urodziny. Aby zobaczyć związek, niech y oznacza urodziny i-tej osoby w grupie. Jeżeli w grupie jest q osób to mamy q wartości y_1, \dots, y_q wybrane jednoznacznie z $\{1, \dots, 365\}$, przyjmując upraszczające założenie, że urodziny są równomiernie i niezależnie rozłożone na 365 dni roku nieprzestępnego. Co więcej, pasujące daty urodzin odpowiadają kolizji, tj. odrębny i, j z $y_i = y_j$. Zatem pożąданie rozwiązanie problemu urodzin jest określone przez minimalną (całkowitą) wartość q , dla której $\text{col}(q, 365) = 1/2$. (Odpowiedź może Cię zaskoczyć – wystarczy przyjąć $q = 23$ osoby!)

W tej sekcji udowodnimy dolną i górną granicę $\text{col}(q, N)$. Zebrane razem i podsumowane na wysokim poziomie pokazują, że jeśli $q < N$, to prawdopodobieństwo zderzenia wynosi $\Theta(q^2/N)$; alternatywnie, dla $q = \Theta(N)$ prawdopodobieństwo zderzenia jest stałe.

Górna granicę prawdopodobieństwa kolizji można łatwo uzyskać.

LEMAT A.15 Ustal dodatnią liczbę całkowitą N i powiedz q elementów y_1, \dots, y_q są wybierane równomiernie i niezależnie losowo ze zbioru wielkości N . Wtedy prawdopodobieństwo, że istnieją różne i, j z $y_i = y_j$ wynosi co najwyżej $\frac{q^2}{2N}$. To jest,

$$\text{col}(q, N) \leq \frac{q^2}{2N}.$$

DOWÓD Dowód polega na prostym zastosowaniu więzu unijnego (Twierdzenie A.7). Przypomnijmy, że kolizja oznacza, że istnieją różne i, j z $y_i = y_j$. Niech Coll oznacza zdarzenie zderzenia, a $\text{Col}_{i,j}$ oznacza zdarzenie, w którym $y_i = y_j$. Jest oczywiste, że $\Pr[\text{Col}_{i,j}] = 1/N$ dla dowolnego odrębnego i, j . Co więcej, $\text{Coll} = \bigcup_{i,j} \text{Col}_{i,j}$, co oznacza wielokrotne zastosowanie związku unijnego

$$\begin{aligned} \Pr[\text{Col}] &= \Pr_{i=j} \text{Col}_{i,j} \\ \Pr[\text{Col}_{i,j}] &= \Pr_{j \neq i} \left(\frac{Q}{2} \cdot \frac{1}{N} \right) = \frac{q^2}{2N}. \end{aligned}$$



LEMAT A.16 Ustal dodatnią liczbę całkowitą N i powiedz, że $q = 2N$ elementów y_1, \dots, y_q wybierane są równomiernie i niezależnie losowo ze zbioru wielkości N . Wtedy prawdopodobieństwo, że istnieją różne i, j z $y_i = y_j$ wynosi co najmniej $q(q-1)/4N$.

$\overline{4N}$. W rzeczywistości,

$$\Pr[\text{coll}(q, N)] = 1 - e^{-q(q-1)/2N} \geq \frac{q(q-1)}{4N}.$$

DOWÓD Przypomnijmy, że kolizja oznacza, że istnieją różne i, j z $y_i = y_j$. Niech Coll oznacza to wydarzenie. Niech NoColl_i będzie zdarzeniem, w którym nie ma kolizji pomiędzy y_1, \dots, y_i , tak ; to znaczy $y_j \neq y_k$ dla wszystkich $j < k \leq i$. Wtedy $\text{NoColl}_q = \text{Coll}$ jest zdarzeniem, w którym w ogóle nie ma kolizji.

Jeśli wystąpi NoColl_q , wówczas dla wszystkich $i \leq q$ musi wystąpić również NoColl_i . Zatem,

$$\Pr[\text{NoColl}_q] = \Pr[\text{NoColl}_1] \cdot \Pr[\text{NoColl}_2 | \text{NoColl}_1] \cdots \Pr[\text{NoColl}_q | \text{NoColl}_{q-1}].$$

Teraz $\Pr[\text{NoColl}_1] = 1$, ponieważ y_1 nie może kolidować ze sobą. Ponadto, jeśli wystąpi zdarzenie NoColl_i , wówczas $\{y_1, \dots, y_i\}$ zawiera i różne wartości; zatem prawdopodobieństwo, że y_{i+1} zderzy się z jedną z tych wartości wynosi, a zatem prawdopodobieństwo, że y_{i+1} nie koliduje z żadną z tych wartości wynosi $\frac{1}{N}$. To znaczy

$$\Pr[\text{NoColl}_{i+1} | \text{NoColl}_i] = 1 - \frac{1}{N},$$

a więc

$$\Pr[\text{NoColl}_q] = \prod_{j=1}^{q-1} \left(1 - \frac{1}{N}\right).$$

Ponieważ $i/N < 1$ dla wszystkich i , mamy $1 - \frac{1}{N} \geq i/N$ (z nierównością A.3) i tak

$$\Pr[\text{NoColl}_q] = \prod_{j=1}^{q-1} \left(1 - \frac{i/N}{N}\right) = np \geq \prod_{j=1}^{q-1} \left(\frac{i/N}{N}\right) = np \geq \frac{q(q-1)}{4N}.$$

Dochodzimy do wniosku, że

$$\Pr[\text{Coll}] = 1 - \Pr[\text{NoColl}_q] = 1 - e^{-q(q-1)/2N} \geq \frac{q(q-1)}{4N},$$

korzystając z nierówności A.4 w ostatnim kroku (zauważ, że $q(q-1)/2N < 1$). ■

A.5 *Pola skończone

W książce używamy pól skończonych bardzo oszczędnie, ale dla zapewnienia kompletności załączamy definicję i kilka podstawowych faktów. Dalsze szczegóły można znaleźć w każdym podręczniku algebra abstrakcyjnej.

DEFINICJA A.17 Pole (skończone) to (skończony) zbiór F wraz z dwiema operacjami binarnymi $+$, \cdot , dla których zachodzi zasada:

- F jest grupą abelową w odniesieniu do operacji „ $+$ ”. Niech 0 oznacza element tożsamości tej grupy.
- $F \setminus \{0\}$ jest grupą abelową w odniesieniu do operacji „ \cdot ”. Pozwalamy 1 oznaczając element tożsamości tej grupy.
Jak zwykle często piszemy ab zamiast $a \cdot b$.
- (Rozdzielność:) Dla wszystkich $a, b, c \in F$ mamy $a \cdot (b + c) = ab + ac$.

Dodatek odwrotny $\in F$, oznaczony przez $-a$, jest unikalnym elementem spełniającym $a + (-a) = 0$; zamiast $b+(-a)$ piszemy $b - a$. Multiplikatywna odwrotność $a \in F \setminus \{0\}$, oznaczona przez a^{-1} jest unikalnym elementem spełniającym $aa^{-1} = 1$; często piszemy b/a zamiast ba^{-1} .

Przykład A.18 Z

wyników rozdziału 8.1.4 wynika, że dla dowolnej liczby pierwszej p zbiór $\{0, \dots, p-1\}$ jest ciałem skończonym w zakresie dodawania i mnożenia modulo p . Pole to oznaczamy przez F_p .

Pola skończone mają bogatą teorię. Do naszych celów potrzebujemy tylko kilku podstawowych faktów. Rząd F to liczba elementów w F (zakładając, że jest skończona). Przypomnijmy również, że q jest potęgą pierwszą, jeśli $q = p^r$ dla pewnej liczby pierwszej p i liczby całkowitej $r \geq 1$.

TWIERDZENIE A.19 Jeżeli F jest ciałem skończonym, to rząd F jest potęgą pierwszą. I odwrotnie, dla każdej potęgi pierwszej q istnieje skończone pole rzędu q , które jest zresztą jedynym takim polem (aż do ponownego oznakowania elementów).

Dla $q = p^r$ z p prime, niech F_q oznacza (unikalne) pole rzędu q . Nazywamy p charakterystyką F_q . Powyższe twierdzenie mówi nam, że cechą każdego ciała skońченego jest liczba pierwsza.

Podobnie jak w przypadku grup, jeśli n jest liczbą całkowitą dodatnią i $a \in F$ to

$$n \cdot a \stackrel{\text{def}}{=} \underbrace{a + \dots + a}_{n \text{ razy}} \quad \text{ i } a \stackrel{\text{def}}{=} \underbrace{-a - \dots - a}_{n \text{ razy}}.$$

Zapis jest rozszerzany dla $n = 0$ w sposób naturalny.

TWIERDZENIE A.20 Niech F_q będzie skończonym ciałem o charakterystyce p . Wtedy dla każdego $a \in F_q$ mamy $p \cdot a = 0$.

Niech $q = p^r$ z p liczbą pierwszą. Dla $r = 1$ w przykładzie A.18 widzieliśmy, że $F_q = F_p$ można przyjąć jako zbiór $\{0, \dots, p-1\}$ przy dodawaniu i mnożeniu

moduł str. Przestrzegamy jednak, że dla $r > 1$ zbiór $\{0, \dots, q-1\}$ nie jest całem podlegającym dodawaniu i mnożeniu modulo q . Na przykład, jeśli przyjmiemy $q = 32 = 9$, to element 3 nie ma multiplikatywnej odwrotności modulo 9.

Skończone ciała o charakterystyce p można przedstawić za pomocą wielomianów na F_p . Podajemy przykład, aby zademonstrować smak konstrukcji, bez omawiania, dlaczego konstrukcja działa lub opisywania ogólnego przypadku.

Konstruujemy pole F_4 , pracując z wielomianami nad F_2 . Napraw wielomian $r(x) = x^2 + x + 1$ i zauważ, że $r(x)$ nie ma pierwiastków w F_2 , ponieważ $r(0) = r(1) = 1$ (pamiętaj, że pracujemy w F_2 , co oznacza aby wszystkie operacje były wykonywane modulo 2). W ten sam sposób możemy wprowadzić liczbę urojoną ω jako pierwiastek z $r(x)$. W ten sposób, aby móc wprowadzić wartość ω jako pierwiastek z $r(x)$ przez F_2 ; to znaczy $\omega = \sqrt{-1}$. Następnie definiujemy F_4 jako zbiór wszystkich wielomianów stopnia 1 w F_2 ; to znaczy $F_4 = \{0, 1, \omega, \omega + 1\}$. Dodawanie w F_4 będzie po prostu zwykłym dodawaniem wielomianów, pamiętając, że operacje na współczynnikach wykonywane są w F_2 (czyli modulo 2). Mnożenie w F_4 będzie mnożeniem wielomianu (znowu z operacjami na współczynnikach przeprowadzanymi modulo 2), po którym następuje podstawienie $\omega = \sqrt{-1}$; zapewnia to również, że wynik leży w F_4 . Więc na przykład

$$\omega + (\omega + 1) = 2\omega + 1 = 1$$

I

$$(\omega + 1) \cdot (\omega + 1) = \omega^2 + 2\omega + 1 = (-\omega - 1) + 1 = \omega = \omega$$

Chociaż nie jest to oczywiste, można sprawdzić, czy jest to pole; jedynym trudnym warunkiem do sprawdzenia jest to, że każdy niezerowy element ma odwrotność multiplikatywną.

Potrzebujemy jeszcze tylko jednego wyniku.

TWIERDZENIE A.21 Niech F_q będzie skończonym ciałem rzędu q . Wtedy grupa abelowa $F_q \setminus \{0\}$ ze względu na „.” jest grupą cykliczną rzędu $q-1$.

Załącznik B

Podstawowa algorytmiczna teoria liczb

Aby konstrukcje kryptograficzne podane w tej książce były efektywne (tj. działały w czasie wielomianem na długościach ich danych wejściowych), konieczne jest, aby te konstrukcje wykorzystywały wydajne (tj. wielomianowe) algorytmy do wykonywania podstawowych operacji liczbowych. operacje teoretyczne. Chociaż w niektórych przypadkach istnieją „trywialne” algorytmy, które zadziałyłyby, nadal warto dokładnie rozważyć ich wydajność, ponieważ w zastosowaniach kryptograficznych nierazko używa się liczb całkowitych o długości tysięcy bitów. W innych przypadkach otrzymanie dowolnego algorytmu czasu wielomianowego wymaga odrobinę sprytu, a analiza ich działania może opierać się na nietrywialnych wynikach teorii grup.

W dodatku B.1 opisujemy podstawowe algorytmy arytmetyki liczb całkowitych. Omówimy tutaj znane algorytmy dodawania, odejmowania itp., a także algorytm Euklidesa do obliczania największych wspólnych dzielników. Omawiamy także rozszerzony algorytm Euklidesa, zakładając, że czytelnik przerobił materiał z rozdziału 8.1.1.

W dodatku B.2 pokazujemy różne algorytmy arytmetyki modułowej. Oprócz krótkiego omówienia podstawowych operacji modułowych (tj. redukcji modułowej, dodawania, mnożenia i inwersji), opisujemy także mnożenie Montgomery'ego, które może znacznie uprościć (i przyspieszyć) implementacje arytmetyki modułowej. Następnie omawiamy algorytmy dla problemów mniej powszechnych poza dziedziną kriptografii: potęgowanie modulo N (jak również w dowolnych grupach) i wybieranie jednolitego elementu ZN lub Z (lub w dowolnej grupie). W tej sekcji założono znajomość podstawowej teorii grup omówionej w sekcji 8.1.

Powyższy materiał jest używany pośrednio przez całą drugą połowę książki, chociaż przeczytanie tego materiału nie jest absolutnie konieczne, aby móc śledzić książkę. (W szczególności czytelnik chcący zaakceptować wyniki tego Dodatku bez dowodu może po prostu przeczytać podsumowanie tych wyników w poniższych twierdzeniach.) Dodatek B.3, który omawia znajdowanie generatorów w grupach cyklicznych (kiedy faktoryzacja rzędu grupowego jest znany) i zakłada wyniki z sekcji 8.3.1, zawiera materiał, który jest w ogóle rzadko używany; jest on dołączony w celu zapewnienia kompletności i odniesienia.

Ponieważ naszym celem jest jedynie ustalenie, że pewne problemy można rozwiązać w czasie wielomianowym, w wyborze algorytmów i ich opisów postawiliśmy raczej na prostotę niż na efektywność (o ile algorytmy działają w czasie wielomianowym). Z tego powodu na ogół nie będziemy zainteresowani

przedstawiamy dokładne czasy działania algorytmów, wykraczające poza ustalenie, że rzeczywiście działają one w czasie wielomianowym. Czytelnika poważnie zainteresowanego implementacją tych algorytmów ostrzega się, aby szukał w innych źródłach bardziej wydajnych alternatyw, a także różnych technik przyspieszających niezbędne obliczenia.

Wyniki zawarte w tym dodatku podsumowano w poniższych twierdzeniach. W całym tekście zakładamy, że każda liczba całkowita a podana jako dane wejściowe jest zapisywana przy użyciu dokładnie bitów; tj. bitem wyższego rzędu jest 1. W dodatku B.1 pokazujemy:

TWIERDZENIE B.1 (Działania na liczbach całkowitych) Mając liczby całkowite aib, można wykonać następujące operacje na wielomianach czasu w aib:

1. Obliczanie sumy $a + b$ i różnicy $a - b$;
2. Obliczanie iloczynu ab ;
3. Obliczanie dodatnich liczb całkowitych q i $r < b$ takich, że $a = qb + r$ (tzn. obliczanie dzielenia z resztą);
4. Obliczanie największego wspólnego dzielnika aib, $\text{gcd}(a, b)$;
5. Obliczanie liczb całkowitych X, Y z $Xa + Yb = \text{gcd}(a, b)$.

Poniższe wyniki przedstawiono w Załączniku B.2:

TWIERDZENIE B.2 (Operacje modułowe) Mając liczby całkowite $N > 1$, a i b, można wykonać następujące operacje na wielomianach czasu w a, b i N:

1. Obliczanie redukcji modułowej $[a \bmod N]$;
2. Obliczenie sumy $[(a+b) \bmod N]$, różnicy $[(a-b) \bmod N]$ i produktu $[ab \bmod N]$;
3. Ustalenie, czy a jest odwracalne modulo N;
4. Obliczanie odwrotności multiplikatywnej $[a^{-1} \bmod N]$, zakładając, że a jest w-pionowo modulo N;
5. Obliczanie potęgowania $[a^b \bmod N]$.

Poniższy uogólnia Twierdzenie B.2(5) na dowolne grupy:

TWIERDZENIE B.3 (Potęgowanie grup) Niech G będzie grupą zapisaną multiplikatywnie. Niech g będzie elementem grupy i niech b będzie nieujemną liczbą całkowitą. Następnie G^b można obliczyć za pomocą operacji na grupach wielo(b).

TWIERDZENIE B.4 (Wybór elementów jednorodnych) Istnieje algorytm losowy o następujących właściwościach: na wejściu N ,

- Algorytm działa w wielomianu czasu w N ;
- Wyniki algorytmu kończą się niepowodzeniem z prawdopodobieństwem znikomym w N ; I
- Pod warunkiem, że wyjście nie zakończy się niepowodzeniem, algorytm wyprowadza równomiernie rozproszony element ZN.

Algorytm o analogicznych właściwościach istnieje dla Z_N również.

Ponieważ prawdopodobieństwo, że którykolwiek z algorytmów, o których mowa w powyższym twierdzeniu, nie powiedzie się, jest znikome, ignorujemy tę możliwość (i zamiast tego pozostawiamy ją ukrytą). W Załączniku B.2 omawiamy także uogólnienia powyższego na przypadek wyboru elementu jednolitego z dowolnej grupy skończonej (z zastrzeżeniem pewnych wymagań dotyczących reprezentacji elementów grupowych).

Dowód poniższego znajduje się w Załączniku B.3:

TWIERDZENIE B.5 (Testowanie i znajdowanie generatorów) Niech G będzie grupą cykliczną rzędu q i założymy, że działanie grupowe i wybór jednolitego elementu grupy można przeprowadzić w jednostce czasu.

1. Istnieje algorytm, który na wejściu q , rozkład na czynniki pierwsze q i elementu g G , działa w czasie $\text{pol}(q)$ i decyduje, czy g jest generatorem G .
2. Istnieje losowy algorytm, który na wejściu q i rozłożeniu na czynniki pierwsze q działa w czasie $\text{pol}(q)$ i generuje generator G , z wyjątkiem tego, że prawdopodobieństwo jest znikome w q . Uwarunkowany tym, że wyjście jest generatorem, jest równomiernie rozdzielony pomiędzy generatorami G .

B.1 Arytmetyka liczb całkowitych

B.1.1 Podstawowe operacje

Rozpoczynamy naszą eksplorację algorytmicznej teorii liczb od omówienia dodawania/odejmowania liczb całkowitych, mnożenia i dzielenia z resztą.

Krótkie przemyślenie pokazuje, że wszystkie te operacje można przeprowadzić na wielomianu czasu na długości wejściowej, stosując standardowe algorytmy „szkolne” dla tych problemów. Na przykład dodanie dwóch dodatnich liczb całkowitych aib z $a > b$ można wykonać w czasie liniowym w a , przechodząc jeden po drugim przez bity aib , zaczynając od bitów mniej znaczących i obliczając kor - odpowiadający bit wyjściowy i „bit przeniesienia” na każdym kroku. (Szczegóły zostały pominięte.)

Mnożenie dwóch n-bitowych liczb całkowitych aib , aby wziąć inny przykład, może

można to zrobić, najpierw generując listę n liczb całkowitych o długości co najwyżej $2n$ (z których każda jest równa $a^{i-1} \cdot b$, gdzie i jest i-tym bitem b), a następnie dodając te n liczb całkowitych, aby otrzymać wynik końcowy. (Dzielenie z resztą jest trudniejsze do skutecznego wdrożenia, ale można to również zrobić.)

Chociaż te algorytmy dla szkół podstawowych wystarczą do wykazania, że powyższe problemy można rozwiązać w czasie wielomianowym, warto zauważać, że algorytmy te w niektórych przypadkach nie są najlepszymi dostępnymi.

Przykładowo podany powyżej prosty algorytm mnożenia mnoży dwie liczby n-bitowe w czasie $O(n)$ w czasie $O(n \log 2^3)$ (a^{2^n}), ale istnieje lepszy działający algorytm nawet to nie jest najlepsze z możliwych). Chociaż różnica jest nieistotna dla liczb wielkości, z jaką spotykamy się na co dzień, staje się to zauważalne, gdy liczby są duże. W zastosowaniach kryptograficznych nierzadko używa się liczb całkowitych o długości tysięcy bitów (tj. $n > 1000$) i rozsądny wybór algorytmów, których należy wtedy użyć staje się krytyczny.

B.1.2 Algorytmy euklidesowe i rozszerzone algorytmy euklidesowe

Przypomnijmy sobie z podrozdziału 8.1, że $\gcd(a, b)$, największy wspólny dzielnik dwóch liczb całkowitych a i b , jest największą liczbą całkowitą d , która dzieli zarówno a , jak i b .

Przedstawiamy proste twierdzenie dotyczące największego wspólnego dzielnika, a następnie pokazujemy, jak prowadzi to do wydajnego algorytmu obliczania \gcd .

TWIERDZENIE B.6 Niech $a, b > 1$ z $b \mid A$. Następnie

$$\gcd(a, b) = \gcd(b, [a \bmod b]).$$

DOWÓD Jeśli $b > a$ stwierdzone roszczenie jest natychmiastowe. Założymy więc, że $a > b$.

Zapisz $a = qb + r$ dla q , r dodatnich liczb całkowitych i $r < b$ (por. Twierdzenie 8.1); zauważ, że $r > 0$, ponieważ $b \mid A$. Ponieważ $r = [a \bmod b]$, dowodzimy twierdzenia pokazując, że $\gcd(a, b) = \gcd(b, r)$.

Niech $d = \gcd(a, b)$. Następnie d dzieli zarówno a i b , więc d dzieli także $r = a - qb$. Z definicji największego wspólnego dzielnika mamy zatem $\gcd(b, r) = d = \gcd(a, b)$.

Niech $d = \gcd(b, r)$. Następnie d dzieli zarówno b , jak i r , więc d dzieli także $a = qb + r$. Z definicji największego wspólnego dzielnika mamy zatem $\gcd(a, b) = d = \gcd(b, r)$.

Ponieważ $d \mid d$ i $d \mid d$, wnioskujemy, że $d = d$. ■

Powyższe sugeruje rekurencyjny algorytm euklidesowy (Algorytm B.7) do obliczania największego wspólnego dzielnika $\gcd(a, b)$ dwóch liczb całkowitych a i b . Poprawność algorytmu wynika łatwo z Twierdzenia B.6. Jeśli chodzi o czas działania, poniżej pokazujemy, że na wejściu (a, b) algorytm wykonuje mniej niż $2 \cdot b$ wywołań rekurencyjnych. Od sprawdzenia, czy b dzieli a i obliczenia

ALGORYTM B.7 Algorytm euklidesowy GCD

Dane wejściowe: liczby całkowite a, b , gdzie $a - b > 0$

Wynik: Największy wspólny dzielnik $a|b$

jeśli b dzieli,

zwróć b, w

przeciwnym razie zwróć $\text{GCD}(b, [a \bmod b])$

$[a \bmod b]$ można wykonać w wielomianach czasu $w_{a|b}$, co oznacza, że cały algorytm działa w czasie wielomianowym.

TWIERDZENIE B.8 Rozważmy wykonanie $\text{GCD}(a_0, b_0)$ i niech a_i (dla $i = 1, \dots, r$) oznacza argumenty i -tego rekurencyjnego wywołania NWD. Następnie $b_{i+2} = b_i/2$ dla $0 \leq i \leq r-2$.

DOWÓD Najpierw zauważmy, że dla dowolnego $a > b$ mamy $[a \bmod b] < a/2$. Aby to zobaczyć, rozważmy dwa przypadki: Jeśli $b \geq a/2$, to $[a \bmod b] < b \geq a/2$ jest natychmiastowe. Z drugiej strony, jeśli $b > a/2$ to $[a \bmod b] = a - b < a/2$.

Teraz napraw dowolne i za pomocą $0 \leq i \leq r-2$. Następnie $b_{i+2} = [a_{i+1} \bmod b_{i+1}] < a_{i+1}/2 = b_i/2$. ■

WNIOSEK B.9 Podczas wykonywania algorytmu $\text{GCD}(a, b)$ występuje co najwyżej $2r-2$ rekurencyjne wywołania GCD.

DOWÓD Niech a_i i b_i (dla $i = 1, \dots, r$) oznacza argumenty i -tego rekurencyjnego wywołania NWD. $\{b_i\}$ są zawsze większe od zera i algorytm nie wykonuje dalszych wywołań rekurencyjnych, jeśli kiedykolwiek zdarzy się, że $b_i = 1$ (od tego momentu $b_i | a_i$). Poprzednie twierdzenie wskazuje, że $\{b_i\}$ zmniejsza się o współczynnik mnożnikowy wynoszący (co najmniej) 2 w każdych dwóch iteracjach. Wynika z tego, że liczba rekurencyjnych wywołań GCD wynosi co najwyżej $2 \cdot (b - 1)$. ■

Rozszerzony algorytm euklidesowy

Z Twierdzenia 8.2 wiemy, że dla dodatnich liczb całkowitych a, b istnieją liczby całkowite X, Y , gdzie $Xa + Yb = \text{gcd}(a, b)$. Prostą modyfikację algorytmu Euklidesa, zwaną rozszerzonym algorytmem Euklidesa, można zastosować do znalezienia X, Y oprócz obliczenia $\text{gcd}(a, b)$; patrz Algorytm B.10. Zostaniesz poproszony o wykazanie poprawności rozszerzonego algorytmu Euklidesa w ćwiczeniu B.1 oraz udowodnienie, że algorytm działa w czasie wielomianowym w ćwiczeniu B.2.

ALGORYTM B.10**Rozszerzony algorytm euklidesowy eGCD**

Dane wejściowe: liczby całkowite a, b , gdzie $a - b > 0$

Wynik: (d, X, Y) gdzie $d = \gcd(a, b)$ i $Xa + Yb = d$

jeśli b dzieli a

powrót $(b, 0, 1)$

Oblicz liczby całkowite q, r gdzie $a = qb + r$ i $0 < r < b$

$(d, X, Y) := eGCD(b, r)$ // zauważ, że $Xb + Yr = d$

powrót $(d, Y, X - Yq)$

B.2 Arytmetyka modułowa

Skupmy się teraz na podstawowych operacjach arytmetycznych modulo $N > 1$.
Będziemy używać ZN w odniesieniu do zbioru $\{0, \dots, N - 1\}$ oraz do grupy wynika to z rozważenia dodania modulo N wśród elementów tego zbioru.

B.2.1 Podstawowe operacje

Efektywne algorytmy podstawowych działań arytmetycznych na liczbach całkowitych natychmiast implikują wydajne algorytmy dla odpowiednich operacji arytmetycznych modulo N. Na przykład obliczenie redukcji modułowej $[a \bmod N]$ można to zrobić w wielomianach czasu w a i N, obliczając dzielenie z resztą przez liczby całkowite. Następnie rozważmy operacje modułowe na dwóch elementach $a, b \in \text{ZN}$ gdzie $N = n$. (Zauważ, że a, b mają długość co najwyżej n. W rzeczywistości wygodnie jest po prostu założyć, że wszystkie elementy ZN mają długość dokładnie n, w razie potrzeby dodając zera w lewo.) Dodanie $a \bmod N$ można wykonać, obliczając najpierw $a+b$, liczbę całkowitą o długości co najwyżej $n+1$, a następnie redukując ten wynik pośredni modulo N. Podobnie, mnożenie modulo N można wykonać, obliczając najpierw liczbę całkowitą ab o długości co najwyżej $2n$, a następnie redukując wynik modulo N. Ponieważ dodawanie, mnożenie i dzielenie z resztą można wykonać w czasie wielomianowym, te podać wielomianowe algorytmy dodawania i mnożenia modulo N.

B.2.2 Obliczanie odwrotności modułowych

Nasza dotychczasowa dyskusja pokazała, jak dodawać, odejmować i mnożyć modulo N. Jedną z operacji, której nam brakuje, jest „dzielenie” lub równoważnie obliczanie Odwrotność multiplikatywna modulo N. Przypomnijmy z podrozdziału 8.1.2, że odwrotność multiplikatywna (modulo N) elementu $a \in \text{ZN}$ jest elementem $a^{-1} \in \text{ZN}$ tak, że $a \cdot a^{-1} \equiv 1 \pmod{N}$. Twierdzenie 8.7 pokazuje, że a ma odwrotność wtedy i tylko wtedy, gdy $\gcd(a, N) = 1$, tj. wtedy i tylko wtedy, gdy $a \in \text{ZN}^*$. Zatem korzystając z

Algorytmem Euklidesa możemy łatwo określić, czy dany element ma multiplikatywną odwrotność modulo N.

Biorąc pod uwagę $N \in \mathbb{Z}$ z $\gcd(a, N) = 1$, Twierdzenie 8.2 mówi nam, że istnieją liczby całkowite X, Y gdzie $Xa + YN = 1$. Oznacza to, że $[X \bmod N]$ jest multiplikatywną odwrotnością a. Liczby całkowite X i Y spełniające $Xa + YN = 1$ można skutecznie znaleźć, korzystając z pokazanego rozszerzonego algorytmu eGCD w sekcji B.1.2. Prowadzi to do następującego algorytmu czasu wielomianowego dla obliczanie odwrotności multiplikatywnych:

ALGORYTM B.11

Obliczanie odwrotności modułowych

Wejście: moduł N; element a

Dane wyjściowe: $[a^{-1} \bmod N]$ (jeśli istnieje)

$(d, X, Y) := \text{eGCD}(a, N) //$ zauważ, że $Xa + YN = \gcd(a, N)$

jeśli $d = 1$ zwróć „a nie jest odwracalne modulo N”

w przeciwnym razie zwróć $[X \bmod N]$

B.2.3 Potęgowanie modułowe

Trudniejszym zadaniem jest potęgowanie modulo N, czyli obliczanie $[a^b \bmod N]$ dla podstawy $a \in \mathbb{Z}$ i wykładnika całkowitego $b > 0$. (Gdy $b = 0$ problem jest łatwy. Gdy $b < 0$ i $a \in \mathbb{Z}_N$, potem $a^b = (a^{-1})^{-b} \bmod N$) i problem sprowadza się do przypadku potęgowania z liczbą dodatnią wykładnik, biorąc pod uwagę, że możemy obliczyć odwrotności, jak omówiono w poprzednim (sekcja.) Zauważ, że podstawowe podejście stosowane w przypadku dodawania i mnożenie (tj. obliczanie liczby całkowitej a^b a następnie zmniejszając tą długość wyniku pośredni modulo N) nie działa tutaj: liczba całkowita a $\bmod N$ ma $\Theta(\log a^b) = \Theta(b \cdot \log a)$, a więc nawet zapisanie wyniku pośredniego a $\bmod N$ wymagałoby wykładniczego czasu w b = $\Theta(\log b)$.

Możemy rozwiązać ten problem, zmniejszając modulo N na wszystkich poziomach pośrednich etapach obliczeń, a nie tylko zmniejszanie modulo N na końcu.

Dzięki temu wyniki pośrednie są przez cały czas „małe”. obliczenia. Nawet mając tę ważną początkową obserwację, zaprojektowanie algorytmu czasu wielomianowego dla potęgowania modułowego nadal nie jest trywialne. Rozważmy naiwne podejście algorytmu B.12, które po prostu wykonuje b mnożenie przez a. To nadal działa w czasie wykładniczym w b.

Ten naiwny algorytm można postrzegać jako oparty na następującej powtarzalności:

$$[A^B \bmod N] = [a \cdot a^{b-1} \bmod N] = [a \cdot a \cdot a^{b-2} \bmod N] = \dots$$

Każdy algorytm oparty na tej zależności będzie wymagał czasu $\Theta(b)$. Możemy zrobić

ALGORYTM B.12 Naiwny
algorytm potęgowania modułowego

Wejście: moduł N ; podstawa $a \in \mathbb{Z}_N$; wykładnik całkowity $b > 0$
Wyjście: $[a \bmod N]^b$

$x := 1$
dla $i = 1$ do b :
 $x := [x \cdot a \bmod N]$
zwróć x

lepiej, opierając się na następującej powtarzalności:

$$[A^B \bmod N] = \begin{cases} a \cdot a^{b-2} \bmod N, & \text{gdy } b \text{ jest parzyste} \\ a \cdot a^{b-1} \bmod N, & \text{gdy } b \text{ jest nieparzyste.} \end{cases}$$

Prowadzi to do algorytmu —zwanego z oczywistych powodów „podniesieniem do kwadratu i pomnożeniem” (lub „wielokrotnym podnoszeniem do kwadratu”) —który wymaga jedynie $O(\log b) = O(b)$ modułowych kwadratur/mnożeń; patrz Algorytm B.13. W tym algorytmie długość b zmniejsza się o 1 w każdej iteracji; wynika z tego, że liczba iteracji wynosi b , więc cały algorytm działa w wielomianach czasu w a , b i N . Dokładniej, liczba modułowych kwadratur wynosi dokładnie b , a liczba dodatkowych modułowych mnożeń jest dokładnie równa Hamminga waga b (tj. liczba jedynek w binarnej reprezentacji b). Wyjaśnia to preferencję omówioną w sekcji 8.2.4, aby publiczny wykładnik RSA e miał małą długość/wagę Hamminga.

ALGORYTM B.13 Algorytm
ModExp do wydajnego potęgowania modułowego

Wejście: moduł N ; podstawa $a \in \mathbb{Z}_N$; wykładnik całkowity $b > 0$

Wyjście: $[a \bmod N]^b$

$x := a$
 $t := 1$

// utrzymuj niezmiennik, że odpowiedzią jest $[t \cdot x, \text{ gdy } b > 0] \bmod N$
wykonaj:

jeśli b jest nieparzyste

$t := [t \cdot x \bmod N]$, $b := b - 1$ $x := [x \bmod N]$, $\frac{b}{2} := b/2$ return t

Ustal a i N i rozważ modułową funkcję potęgowania podaną przez $f_{a,N}(b) = [a \bmod N]^b$. Właśnie widzieliśmy, że obliczenie $f_{a,N}$ jest łatwe. Natomiast obliczenie odwrotności tej funkcji – to znaczy obliczenie b przy danych a , N i $[a \bmod N]$ – uważa się za trudne w przypadku odpowiedniego wyboru B .

i N. Odwrócenie tej funkcji wymaga rozwiązyania problemu logarytmu dyskretnego, coś, co szczegółowo omawiamy w sekcji 8.3.2.

Korzystanie z obliczeń wstępnych. Jeśli baza a jest znana z góry i istnieje ograniczone długością wykładnika b, wówczas można zastosować obliczenia wstępne i niewielka ilość pamięci, aby przyspieszyć obliczenia $[a^b \bmod N]$. Mówić b n. Następnie wstępnie obliczamy i przechowujemy n wartości

$$x_0 := a, x_1 := [a^2 \bmod N], \dots, x_{n-1} := [a^{2^{n-1}} \bmod N].$$

Dany wykładnik b z reprezentacją binarną $b_n \dots b_0$ (zapisane z most do najmniej znaczącego bitu), wówczas mamy

$$[a^b] = [a^{\sum_{j=0}^{n-1} 2^j \cdot b_j}] = \prod_{j=0}^{n-1} x_j^{b_j} \bmod N.$$

Ponieważ $b_i \in \{0, 1\}$, liczba mnożeń potrzebnych do obliczenia wyniku jest dokładnie o jeden mniejszy niż waga Hamminga b.

Potęgowanie w grupach arbitralnych

Podany powyżej wydajny modułowy algorytm potęgowania ma zastosowanie w prosty sposób na umożliwienie wydajnego potęgowania w dowolnej grupie, o ile ponieważ podstawowa operacja grupowa może być wykonywana efektywnie. Konkretnie, jeśli G jest grupą i g jest elementem G, to g $\overset{B}{\rightarrow}$ można obliczyć za pomocą at większość $2 \cdot b$ zastosowania podstawowej operacji grupowej. Obliczenia wstępne można również zastosować, dokładnie tak, jak opisano powyżej.

Jeśli rząd q G jest znany, to a $\overset{B}{\rightarrow} [a^q \bmod q]$ (por. Twierdzenie 8.52) i można to wykorzystać do przyspieszenia obliczeń, najpierw zmniejszając b modulo q.

Biorąc pod uwagę grupę (addytywną) ZN, algorytm potęgowania grupowego właśnie opisane daje metodę obliczania „potęgowania”

$$[b \cdot g \bmod N] \overset{\text{def}}{=} [\underbrace{sol + \dots + sol}_{\text{razy}} \bmod N]$$

różni się to od metody omówionej wcześniej, która opiera się na standardowej liczbie całkowitej mnożenie, po którym następuje redukcja modułowa. Porównując dwa podejścia do rozwiązania tego samego problemu, należy zauważać, że oryginalny algorytm wykorzystuje szczegółowe informacje o ZN; w szczególności traktuje (zasadniczo) „wykładnik” b jako element ZN (prawdopodobnie najpierw redukując b modulo N). W dla kontrastu, przedstawiony właśnie algorytm „podnieś i pomnóż” traktuje tylko ZN jako grupę abstrakcyjną. (Oczywiście, grupowe działanie dodawania modulo N opiera się na specyfice ZN.) Celem tej dyskusji jest jedynie zilustrowanie, że niektóre algorytmy grupowe są ogólne (tj. mają zastosowanie równie dobrze do wszystkich grup), podczas gdy niektóre algorytmy grupowe opierają się na określonych właściwościach określonej grupy lub klasy grup. Widzieliśmy kilka przykładów tego zjawiska w Rozdziale 9.

B.2.4 *Mnożenie Montgomery'ego

Chociaż dzielenie liczb całkowitych (a co za tym idzie redukcja modułowa) można przeprowadzić w czasie wielomianowym, algorytmy dzielenia liczb całkowitych są powolne w porównaniu z, powiedzmy, algorytmami mnożenia liczb całkowitych. Mnożenie Montgomery'ego umożliwia wykonywanie mnożenia modułowego bez przeprowadzania kosztownych redukcji modułowych. Ponieważ wymagane jest przetwarzanie wstępne i końcowe, metoda ta jest korzystna tylko wtedy, gdy zostanie wykonanych kilka mnożeń modułowych po kolej, jak np. przy obliczaniu potęgowania modułowego.

Ustal nieparzysty moduł N , w odniesieniu do którego mają zostać wykonane operacje modułowe. Niech $R > N$ będzie potęgą dwójki, powiedzmy $R = 2w$ i zauważ, że $\gcd(R, N) = 1$. Kluczową właściwością, którą wykorzystamy, jest to, że dzielenie przez R jest szybkie: iloraz x po dzieleniu przez R uzyskuje się po prostu przesuwając x w prawo pozycję w , a $[x \bmod R]$ to po prostu w najmniej znaczące bity x .

Zdefiniuj reprezentację Montgomery'ego $x \in \mathbb{Z}_N$ przez $\bar{x} \in \mathbb{Z}_N$ definiując $\bar{x} = [xR \bmod N]$. Mnożenie Montgomery'ego przez $\bar{x}, \bar{y} \in \mathbb{Z}_N$ definiuje się jako

$$\text{Mont}(\bar{x}, \bar{y}) \stackrel{\text{def}}{=} [\bar{x}\bar{y}R^{-1} \bmod N].$$

(Pokazujemy poniżej, jak można to obliczyć bez żadnych kosztownych redukcji modułowych.) Zauważ, że

$$\text{Mont}(\bar{x}, \bar{y}) = \bar{x}\bar{y}R^{-1} = (xR)(yR)R^{-1} = (xy)R = xy \bmod N.$$

Oznacza to, że możemy pomnożyć kilka wartości w \mathbb{Z}_N poprzez (1) konwersję do reprezentacji Montgomery'ego, (2) wykonanie wszystkich mnożeń przy użyciu mnożenia Montgomery'ego w celu uzyskania końcowego wyniku, a następnie (3) przekształcenie wyniku z reprezentacji Montgomery'ego z powrotem do standardowej reprezentacji $\bmod N$.

Niech $a \stackrel{\text{def}}{=} [\bar{N}^{-1}R]$, wartość, którą można wstępnie obliczyć. (Obliczenie a i konwersja do/z reprezentacji Montgomery'ego można również wykonać bez żadnych kosztownych redukcji modułowych; szczegóły wykraczają poza nasz zakres.)

obliczyć $c \stackrel{\text{def}}{=} \text{Mont}(x, y)$ bez kosztownych redukcji modułowych:

1. Niech $z := x \cdot y$ (na liczbach całkowitych).

2. Ustaw $c := (z + [za \bmod R] \cdot N) / R$.

3. Jeżeli $c < N$ to ustaw $c := c$; w przeciwnym razie ustaw $c := c - N$.

Aby zobaczyć, że to działa, musimy najpierw sprawdzić, czy krok 2 jest dobrze zdefiniowany, a mianowicie, czy licznik jest podzielny przez R . Dzieje się tak, ponieważ

$$z + [za \bmod R] \cdot N = z + zaN = z - zN - 1N = 0 \bmod R.$$

Następnie zauważ, że $c = z/R \bmod N$ po kroku 2; ponadto, ponieważ $z < N^2 < RN$ mamy $0 < c < (z + RN)/R < 2RN/R = 2N$. Ale wtedy $[c \bmod N] = c$ jeśli $c < N$ i $[c \bmod N] = c - N$ jeśli $c > N$. Dochodzimy do wniosku, że

$$c = [c \bmod N] = [z/R \bmod N] = [xyR^{-1} \bmod N],$$

B.2.5 Wybór jednolitego elementu grupy

W przypadku zastosowań kryptograficznych często konieczne jest wybranie jednolitego elementu grupy G . Najpierw traktujemy problem w ujęciu abstrakcyjnym, a następnie skupiamy się konkretnie na przypadkach ZN i Z .

Należy zauważyć, że jeśli G jest grupą cykliczną rzędu q i znany jest generator $g \in G$, to wybór elementu jednorodnego $h \in G$ sprowadza się do wybrania jednolitej liczby całkowitej $x \in Z_q$ i ustalenia $h := g^x$. W dalszej części nie przyjmujemy żadnych założeń co do G .

Elementy grupy G muszą być określone przy użyciu reprezentacji tych elementów w postaci ciągów bitów, gdzie zakładamy bez utraty ogólności, że wszystkie elementy są reprezentowane za pomocą ciągów znaków o tej samej długości. (Ważne jest również, aby każdy element grupy miał unikalny ciąg znaków.) Na przykład, jeśli $N = n$, wówczas wszystkie elementy ZN można przedstawić jako ciągi o długości n , gdzie liczba całkowita $a \in ZN$ jest dopełniana w lewo za pomocą $0s$ jeśli $a < n$.

Nie skupiamy się zbytnio na zagadnieniu reprezentacji, gdyż dla wszystkich grup rozpatrywanych w tym tekście reprezentację można po prostu przyjąć jako „naturalną” (jak w przypadku ZN powyżej). Należy jednak pamiętać, że różne reprezentacje tej samej grupy mogą wpływać na złożoność wykonywania różnych obliczeń, dlatego wybór „właściwej” reprezentacji dla danej grupy jest często ważny w praktyce. Ponieważ naszym celem jest jedynie pokazanie algorytmów czasu wielomianowego dla każdej potrzebnej operacji (a nie pokazanie najbardziej wydajnych znanych algorytmów), dokładna zastosowana reprezentacja jest dla naszych celów mniej istotna. Co więcej, większość przedstawionych przez nas algorytmów „wyższego poziomu” wykorzystuje operację grupową w sposób „czarnej skrzynki”, tak więc dopóki operację grupową można wykonać w czasie wielomianowym (w jakimś parametrze), wynikowa Algorytm będzie również działał w czasie wielomianowym.

Mając grupę G , w której elementy są reprezentowane przez ciągi o długości n , można wybrać jednolity element grupy, wybierając jednolite ciągi bitowe, aż do znalezienia pierwszego ciągu odpowiadającego elementowi grupy. (Zauważ, że zakłada się, że przynależność do grupy testującej może zostać przeprowadzona efektywnie.) Aby otrzymać algorytm o ograniczonym czasie działania, wprowadzamy parametr t ograniczający maksymalną liczbę powtórzeń tego procesu; jeśli wszystkie iteracje nie znajdą elementu G , wówczas wyniki algorytmu nie powiodą się. (Alternatywą jest wypisanie dowolnego elementu G .) To znaczy:

ALGORYTM B.14 Wybór jednolitego elementu grupy

Wejście: A (opis a) grupy G ; parametr długości ; parametr t

Wyjście: Jednolity element G
dla $i = 1$ do t :

Wybierz uniform $x \in \{0, 1\}$
jeśli $x \in G$ zwróć x

zwróć „niepowodzenie”

Oczywiście jest, że ilekroć powyższy algorytm nie zakończy się niepowodzeniem, generuje element G o równomiernym rozkładzie. Dzieje się tak po prostu dlatego, że każdy element G ma równe prawdopodobieństwo, że zostanie wybrany w dowolnej iteracji. Formalnie, jeśli pozwolimy Fail być zdarzeniem, w którym wyjścia algorytmu zawiodą, to dla dowolnego elementu $g \in G$ mamy

$$\Pr[\text{wynik algorytmu jest równy } g | \text{ Niepowodzenie}] = \frac{1}{|G|}.$$

Jakie jest prawdopodobieństwo, że wyniki algorytmu zawiodą? W dowolnej iteracji prawdopodobieństwo, że $x \in G$ wynosi dokładnie $|G|/2$, a więc prawdopodobieństwo, że x nie należy do G w żadnej z iteracji, wynosi

$$1 - \frac{|G|}{2}^T. \quad (\text{B.1})$$

Istnieje kompromis pomiędzy czasem działania algorytmu B.14 a prawdopodobieństwem, że wyniki algorytmu zawiodą: zwiększenie t zmniejsza prawdopodobieństwo awarii, ale zwiększa czas działania w najgorszym przypadku. Do zastosowań kryptograficznych potrzebny jest algorytm, w którym najgorszy czas działania jest wielomianem w parametrze bezpieczeństwa n , podczas gdy prawdopodobieństwo awarii jest znikome w n .

Niech $K \stackrel{\text{def}}{=} 2/|G|$. Jeśli ustawimy $t := K \cdot n$, to prawdopodobieństwo, że wyjścia algorytmu zawiodą, wynosi:

$$1 - \frac{1}{K}^{K \cdot n} = 1 - \frac{1}{K}^K \cdot \frac{n}{n}^N \approx 1^n = \text{mi}$$

korzystając z Twierdzenia A.2. Zatem, jeśli $K = \text{poli}(n)$ (zakładamy algorytm generowania grup zależny od parametru bezpieczeństwa n , a więc zarówno $|G|$, jak i są funkcjami n), otrzymujemy algorytm o pożądanych właściwościach.

Sprawa Z_N . Rozważmy grupę Z_N , gdzie $n = N$. Sprawdzenie, czy n -bitowy ciąg x (interpretowany jako dodatnia liczba całkowita o długości co najwyżej n) jest elementem Z_N , wymaga po prostu sprawdzenia, czy $x < N$. Ponadto,

$$\frac{2^N}{|Z_N|} = \frac{2^N}{N} \cdot \frac{2^N}{2^n - 1} = 2^n,$$

i w ten sposób możemy próbować jednorodny element Z_N w czasie $\text{poli}(n)$ i z zaniedbywalnym prawdopodobieństwem uszkodzenia w n .

Przypadek Z_N . Rozważmy następnie grupę Z_N , gdzie $n = N$ jak poprzednio. Jest ustalenie, czy n -bitowy ciąg x jest elementem Z (ćwiczenia). Ponadto, n również łatwe (patrz

$$\frac{2^N}{|Z_N|} = \frac{2^N}{\varphi(N)} = \frac{2^N}{N} \cdot \frac{N}{\varphi(N)} = 2^n \cdot \frac{N}{\varphi(N)}.$$

Górna granica $\text{poli}(n)$ jest konsekwencją następującego twierdzenia.

TWIERDZENIE B.15 Dla $N \geq 3$ długości n mamy $\varphi(N) = N - \frac{N}{\varphi(N)} < 2n$.

(Znane są mocniejsze granice, ale powyższe wystarczy dla naszych celów.) Twierdzenie można udowodnić za pomocą Postulatu Bertranda (Twierdzenie 8.32), ale poprostannie na dowodzie w dwóch szczególnych przypadkach: gdy N jest liczbą pierwszą i gdy N jest iloczynem dwóch różnych (różnych) liczb pierwszych.

Analiza jest łatwa, gdy N jest nieparzystą liczbą pierwszą. Tutaj $\varphi(N) = N - 1$ i tak

$$\frac{N}{\varphi(N)} = \frac{2^N}{N-1} = \frac{2^N}{2^n-1} = 2$$

(wykorzystując fakt, że N jest nieparzyste w przypadku drugiej nierówności). Rozważmy następnie przypadek $N = pq$ dla $p \neq q$ różnych, nieparzystych liczb pierwszych. Następnie

$$\frac{N}{\varphi(N)} = \frac{pk}{(p-1)(q-1)} = \frac{P}{p-1} \cdot \frac{Q}{q-1} < \frac{3}{2} \cdot \frac{5}{4} < 2.$$

Dochodzimy do wniosku, że gdy N jest liczbą pierwszą lub iloczynem dwóch różnych, nieparzystych liczb pierwszych, istnieje algorytm generowania N który działa w czasie jednolitego elementu wielomianu Z w $n = N$, a wyniki zawodzą z prawdopodobieństwem znakomitym w n .

Kiedy w tej książce mówimy o próbkowaniu jednolitego elementu, po prostu ignorujemy zakładając N znakome prawdopodobieństwo uzyskania niepowodzenia w przypadku ZN lub Z , że nie ma to znaczącego wpływu na analizę.

B.3 *Wyznaczanie generatora grupy cyklicznej

W tej sekcji zajmiemy się problemem znalezienia generatora dowolnej grupy cyklicznej G rzędu q . Tutaj q niekoniecznie oznacza liczbę pierwszą; w istocie znalezienie generatora, gdy q jest liczbą pierwszą, jest trywialne, zgodnie z wnioskiem 8.55.

Właściwie pokazujemy, jak próbkować generator jednorodny, postępując w sposób bardzo podobny do tego z sekcji B.2.5. Tutaj wielokrotnie próbujemy jednolite elementy G , aż znajdziemy element będący generatorem. Podobnie jak w sekcji B.2.5, analiza tej metody wymaga zrozumienia dwóch rzeczy:

- Jak skutecznie sprawdzić, czy dany element jest generatorem; I
- część elementów grupy, które są generatorami.

Aby zrozumieć te kwestie, najpierw opracujemy dodatkowe tło z teorii grup.

B.3.1 Podstawy teorii grup

Najpierw zajmiemy się drugą kwestią. Przypomnijmy, że rząd elementu h jest najmniejszą dodatnią liczbą całkowitą i dla której $h^l = 1$. Niech g będzie generatorem a

grupa G rzędu $q > 1$; oznacza to, że rząd g to q . Rozważmy element $h \in G$, który nie jest tożsamością (tożsamość nie może być generatorem G) i zapytajmy, czy h może być również generatorem G . Ponieważ g generuje G , my x dla pewnego $x \in \{1, \dots, q-1\}$ (zauważ $x = 0$, ponieważ h nie jest tożsamością, która można zapisać $h = g$). Rozważmy dwa przypadki:

Przypadek 1: $\gcd(x, q) = r > 1$. Zapisz $x = a \cdot r$ i $q = \beta \cdot r + \alpha$, β niezerowymi liczbami całkowitymi mniejszymi niż q . Następnie:

$$\text{gdy } \beta = (g \cdot x)^{\alpha} \quad \beta \cdot \alpha \cdot \beta = g^{\alpha} = (g \cdot q)^{\alpha} = 1.$$

Zatem rząd h wynosi co najwyżej $\beta < q$, a h nie może być generatorem G .

Przypadek 2: $\gcd(x, q) = 1$. Niech $i \in \mathbb{Z}$ będzie rządem h . Następnie

$$0 \equiv a = hg \pmod{q} \quad = (g \cdot x)^i \equiv \text{sol}^{xi},$$

sugerując $xi = 0 \pmod{q}$ zgodnie z Twierdzeniem 8.53. Oznacza to, że $q \mid xi$. Ponieważ jednak $\gcd(x, q) = 1$, Twierdzenie 8.3 pokazuje, że $q \nmid i$, a zatem $i = q$. Dochodzimy do wniosku, że h jest generatorem G .

Podsumowując powyższe, widzimy, że dla $x \in \{1, \dots, q-1\}$ element jest generatorem G dokładnie wtedy, gdy $\gcd(x, q) = 1$. W ten sposób udowodniliśmy $h = g$, co następuje:

TWIERDZENIE B.16 Niech G będzie grupą cykliczną rzędu $q > 1$ z generatorem g .

Istnieją generatory $\varphi(q)$ G i są one dokładnie dane przez $\{g \cdot x \mid x \in \mathbb{Z}_q\}$

W szczególności, jeśli G jest grupą rzędu pierwszego q , to ma ona generatory $\varphi(q) = q - 1$ – dokładnie zgodnie z wnioskiem 8.55.

Przechodzimy dalej do pierwszej kwestii, czyli ustalenia, czy dany element h jest generatorem G . Oczywiście jednym ze sposobów sprawdzenia, czy h generuje G h^1, h^2, \dots, h^{q-1} i zobacz, czy ta lista polega na wyliczeniu $\{h^0, h^1, \dots, h^{q-1}\}$, zawiera każdy element G . Wymaga to czasu liniowego w q (tj. wykładniczego w q) i dlatego jest nie do przyjęcia dla naszych celów. Innym podejściem, jeśli znamy już generator g , jest obliczenie logarytmu dyskretnego $x = \log_g h$, a następnie zastosowanie poprzedniego twierdzenia; ogólnie jednak możemy nie mieć takiego ag , a zresztą obliczenie logarytmu dyskretnego samo w sobie może być trudnym problemem.

Jeśli znamy faktoryzację q , możemy zrobić to lepiej.

TWIERDZENIE B.17 Niech G będzie grupą rzędu q i niech $q = p_1^{e_1} p_2^{e_2} \dots p_n^{e_n}$ będzie rozkładem na czynniki pierwsze q , gdzie $\{p_i\}$ są różnymi liczbami pierwszymi i $e_i \geq 1$.

Ustaw $q_i = q/p_i$. Wtedy $h \in G$ jest generatorem G wtedy i tylko wtedy, gdy

$$\text{gdy } q_i = 1 \text{ dla } i = 1, \dots, n.$$

DOWÓD Jeden kierunek jest łatwy. Powiedzmy $h^{qi} = 1$ dla pewnego i . Wtedy rząd h jest co najwyżej $qi < q$, więc h nie może być generatorem.

I odwrotnie, powiedzmy, że h nie jest generatorem, ale zamiast tego ma rząd $q < q$. Z Twierdzenia 8.54 wiemy, że $q | Q$. Oznacza to, że q można zapisać jako $q = ej$. Ale i tak (używając Twierdzenia 8.53)

$$\begin{aligned} \prod_{i=1}^k p_i^{e_{ia}} \text{ gdzie np } p_i = 0 \text{ dla co najmniej jednego indeksu } j \text{ mamy } e_j h^{qj} \\ \text{mod } q \prod_{i=1}^k p_i^{e_{ia}} = h^q = 1. \end{aligned}$$

= ■

Twierdzenie nie wymaga, aby G było cykliczne; jeśli G nie jest cykliczny, to każdy element $h \in G$ spełni $h^{qi} = 1$ dla pewnego i i nie ma generatorów.

B.3.2 Efektywne algorytmy

Uzbrojeni w wyniki poprzedniej sekcji pokazujemy, jak skutecznie sprawdzić, czy dany element jest generatorem, a także jak sprawnie znaleźć generator w dowolnej grupie.

Testowanie, czy element jest generatorem. Twierdzenie B.17 od razu sugeruje efektywny algorytm decydowania, czy dany element h jest generatorem, czy nie.

ALGORYTM B.18 Sprawdzanie, czy element jest generatorem

Wejście: Kolejność grupowa q ; czynniki pierwsze p_1, p_2, \dots, p_k z q ; element $h \in G$

Wynik: decyzja, czy h jest generatorem G

dla $i = 1$ do k :

jeśli $h^{q/p_i} = 1$ zwróć „ h nie jest generatorem”
zwróć „ h jest generatorem”

Poprawność algorytmu wynika z Twierdzenia B.17. Pokażemy teraz, że algorytm kończy się wielomianem czasu w q . Ponieważ w każdej iteracji h^{q/p_i} można obliczyć w czasie wielomianowym, wystarczy pokazać, że liczba iteracji k jest wielomianowa. Dzieje się tak, ponieważ liczba całkowita q nie może mieć więcej niż $\log_2 q = O(q)$ czynników pierwszych; to dlatego, że

$$q = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k} \quad 2 = 2 \text{ tys}$$

i tak $k \leq \log_2 q$.

Algorytm B.18 wymaga, aby jako dane wejściowe podano czynniki pierwsze rzędu grupowego q . Co ciekawe, nie jest znany skuteczny algorytm sprawdzający, czy element dowolnej grupy jest generatorem, gdy nie są znane czynniki porządku grupowego.

Część elementów będących generatorami. Jak pokazano w Twierdzeniu B.16, część elementów grupy G rzędu q , które są generatorami, wynosi $\varphi(q)/q$. Twierdzenie B.15 mówi, że $\varphi(q)/q = \Omega(1/q)$. Część elementów będących generatorami jest zatem wystarczająco duża, aby zapewnić, że próbkowanie wielomianowej liczby elementów z grupy da generator z prawie znikomym prawdopodobieństwem. (Analiza jest taka sama jak w sekcji B.2.5.)

Konkretnie przykłady w Z , wydajny. Łącząc wszystko w całość, widzimy, że tak algorytm probabilistyczny do znajdowania generatora grupy G , o ile znana jest faktoryzacja rzędu grup. Dlatego przy wyborze grupy do zastosowań kryptograficznych ważne jest, aby grupa została wybrana w taki sposób, aby spełniała te wymagania. To ponownie wyjaśnia preferencję, szczególnie omówioną w sekcji 8.3.2, dotyczącą pracy w odpowiedniej podgrupie Z rzędu pierwszego dla pa silnej liczby pierwszej (tj. $p = 2q+1$ z q również liczbą pierwszą), w którym to przypadku właściwie jestynikipierwsze rząd grupowy $p - 1$ jest znany. Ostatnią możliwością jest wygenerowanie liczby pierwszej p w taki sposób, że znana jest faktoryzacja $p - 1$. Dalsze szczegóły wykraczają poza zakres tej książki.

Referencje i dodatkowe lektury

Książka Shoup [159] jest wysoce zalecana dla tych, którzy chcą bardziej szczegółowo zgłębić tematykę tego rozdziału. W szczególności granice $\varphi(N)/N$ (oraz asymptotyczną wersję Twierdzenia B.15) można znaleźć w [159, Rozdział 5].

Hankerson i in. [83] dostarczają również obszernych szczegółów na temat implementacji algorytmów teorii liczb w kryptografii.

Ćwiczenia

B.1 Wykazać poprawność rozszerzonego algorytmu Euklidesa.

B.2 Udowodnić, że rozszerzony algorytm Euklidesa działa w wielomianu czasu na długościach swoich wejść.

Wskazówka: Najpierw udowodnij twierdzenie analogiczne do twierdzenia B.8.

B.3 Pokaż, jak ustalić, że n -bitowy ciąg znaków znajduje się w Z_N w czasie wielomianowym.

Bibliografia

- [1] M. Abdalla, JH An, M. Bellare i C. Namprempre. Od identyfikacji do podpisów poprzez transformację Fiata-Shamira: warunki konieczne i wystarczające dla bezpieczeństwa i bezpieczeństwa przyszłości. *IEEE Trans. Teoria informacji*, 54(8):3631–3646, 2008.
- [2] M. Abdalla, M. Bellare i P. Rogaway. Założenia wyroczni Diffiego-Hellmana i analiza DHIES. In *Cryptographers' Track – RSA 2001*, tom 2020 LNCS, s. 143–158. Springer, 2001. Zob. <http://cseweb.ucsd.edu/~mihir/papers/dhies.html>.
- [3] C. Adams i S. Lloyd. *Zrozumienie infrastruktury PKI: koncepcje, standardy i kwestie dotyczące wdrożenia*. Addison Wesley, wydanie drugie, 2002.
- [4] LM Adleman. Algorytm subwykładniczy dla problemu logarytmu dyskretnego w zastosowaniach do kriptografii. W 20. dorocznym sympozjum na temat podstaw informatyki, strony 55–60. IEEE, 1979.
- [5] M. Agrawal, N. Kayal i N. Saxena. PRIMES znajduje się w *P*. *Annals of Mathematics*, 160(2):781–793, 2004.
- [6] W. Aiello, M. Bellare, G. Di Crescenzo i R. Venkatesan. Wzmocnienie bezpieczeństwa przez kompozycję: przypadek podwójnie iterowanych, idealnych szyfrów. W *Advances in Cryptology —Crypto '98*, tom 1462 LNCS, strony 390–407. Springer, 1998.
- [7] A. Akavia, S. Goldwasser i S. Safra. Dowodzenie twardych predykatów za pomocą dekodowania list. w proc. 44. doroczne sympozjum na temat podstaw informatyki, s. 146–157. IEEE, 2003.
- [8] W. Alexi, B. Chor, O. Goldreich i CP Schnorr. Funkcje RSA i Rabina: Niektóre części są tak samo twarde jak całość. *SIAM Journal on Computing*, 17 (2): 194–209, 1988.
- [9] NJ AlFardan, DJ Bernstein, KG Paterson, B. Poettering i JCN Schuldt. O bezpieczeństwie RC4 w TLS i WPA. Podczas Sympozjum Bezpieczeństwa USENIX, 2013.
- [10] JH An, Y. Dodis i T. Rabin. O bezpieczeństwie wspólnego podpisu i szyfrowania. W *Advances in Cryptology —Eurocrypt 2002*, tom 2332 LNCS, strony 83–107. Springer, 2002.

- [11] R. Barbulescu, P. Gaudry, A. Joux i E. Thom' e. Heurystyczny algorytm quasi-wielomianowy dla logarytmu dyskretnego w ciałach skończonych o małej charakterystyce. W Advances in Cryptology —Eurocrypt 2014, tom 8441 LNCS, strony 1–16. Wiosna, 2014.
- [12] R. Bardou, R. Focardi, Y. Kawamoto, L. Simionato, G. Steel i J.-K. Tsay. Skuteczne dopełnianie ataków Oracle na sprzęt kryptograficzny. W Advances in Cryptology —Crypto 2012, tom 7417 LNCS, strony 608–625. Springer, 2012.
- [13] E. Barker, W. Barker, W. Burr, W. Polk i M. Smid. Rekomendacja dla kluczowej kadry kierowniczej – część 1: Ogólne (wersja 3), lipiec 2012. Publikacja specjalna NIST 800-57.
- [14] M. Bellare, R. Canetti i H. Krawczyk. Kluczowanie funkcji skrótu do uwierzytelniania wiadomości. W Advances in Cryptology —Crypto '96, tom 1109 LNCS, strony 1–15. Springer, 1996.
- [15] M. Bellare, A. Desai, E. Jokipii i P. Rogaway. Konkretne zabezpieczenie szyfrowania symetrycznego. w proc. 38. doroczne sympozjum na temat podstaw informatyki, strony 394–403. IEEE, 1997.
- [16] M. Bellare, A. Desai, D. Pointcheval i P. Rogaway. Relacje między pojęciami bezpieczeństwa w schematach szyfrowania klucza publicznego. W Advances in Cryptology —Crypto '98, tom 1462 LNCS, strony 26–45. Springer, 1998.
- [17] M. Bellare, O. Goldreich i A. Mityagin. Siła zapytań weryfikacyjnych w uwierzytelnianiu wiadomości i uwierzytelnionym szyfrowaniu. Dostępne pod adresem <http://eprint.iacr.org/2004/309>.
- [18] M. Bellare, J. Kilian i P. Rogaway. Bezpieczeństwo kodu uwierzytelniającego wiadomości łączącego bloki szyfru. Journal of Computer and System Sciences, 61 (3): 362–399, 2000.
- [19] M. Bellare i C. Namprempre. Szyfrowanie uwierzytelnione: Relacje między pojęciami i analiza paradygmatu kompozycji gatunkowej. W Advances in Cryptology —Asiacrypt 2000, tom 1976 LNCS, strony 531–545. Springer, 2000.
- [20] M. Bellare i G. Neven. Podpisy wielokrotne w zwykłym modelu klucza publicznego i ogólny lemat o rozwidleniu. W 13. konferencji ACM. na temat bezpieczeństwa komputerów i komunikacji, strony 390–399. Prasa ACM, 2006.
- [21] M. Bellare i P. Rogaway. Losowe wyrocznie są praktyczne: paradygmat projektowania wydajnych protokołów. W pierwszej konferencji ACM. w sprawie bezpieczeństwa komputerów i komunikacji, strony 62–73. AKM, 1993.

- [22] M. Bellare i P. Rogaway. Optymalne szyfrowanie asymetryczne. W *Ad-vances in Cryptology —Eurocrypt '94*, tom 950 LNCS, strony 92–111. Springer, 1994.
- [23] M. Bellare i P. Rogaway. Dokładne bezpieczeństwo podpisów cyfrowych: jak podpisywać w RSA i Rabin. W *Advances in Cryptology —Eurocrypt '96*, tom 1070 LNCS, strony 399–416. Springer, 1996.
- [24] M. Bellare i P. Rogaway. Bezpieczeństwo potrójnego szyfrowania i ramy dla dowodów gier opartych na kodzie. W *Advances in Cryptology —Eurocrypt 2006*, tom 4004 LNCS, strony 409–426. Springer, 2006. Pełna wersja artykułu jest dostępna pod adresem <http://eprint.iacr.org/>.
- [25] SM Bellovin. Frank Miller: wynalazca jednorazowego notesu. *Kryptologia*, 35(3):203–222, 2011.
- [26] DJ Bernstein. Krótki dowód na nieprzewidywalność łączenia bloków szyfru. Dostępne pod adresem <http://cr.yp.to/papers.html#easycbc>.
- [27] DJ Bernstein. Jak rozciągać funkcje losowe: Bezpieczeństwo chronionych sum licznikowych. *Journal of Cryptology*, 12 (3): 185–192, 1999.
- [28] G. Bertoni, J. Daemen, M. Peeters i G. Van Assche. O niezróżnicowalności budowy gąbek. W *Advances in Cryptology —Eurocrypt 2008*, tom 4965 LNCS, strony 181–197. Springer, 2008.
- [29] E. Biham i A. Shamir. Kryptanaliza różnicowa kryptosystemów typu DES. *Journal of Cryptology*, 4 (1): 3–72, 1991.
- [30] E. Biham i A. Shamir. Różnicowa kryptoanaliza danych standardu szyfrowania. Springer, 1993.
- [31] A. Biriukow i A. Szamir. Kryptanaliza strukturalna SASAS. *J. Kryptologia*, 23(4):505–518, 2010.
- [32] J. Black i P. Rogaway. MAC CBC dla wiadomości o dowolnej długości: konstrukcje trzech kluczy. *Journal of Cryptology*, 18 (2): 111–131, 2005.
- [33] J. Black, P. Rogaway, T. Shrimpton i M. Stam. Analiza funkcji skrótu opartych na szyfrach blokowych z PGV. *J. Kryptologia*, 23(4):519–545, 2010.
- [34] D. Bleichenbachera. Wybrane ataki szyfrogramem na protokoły oparte na standardzie szyfrowania RSA PKCS#1. W *Advances in Cryptology —Crypto '98*, tom 1462 notatek z wykładów z informatyki, strony 1–12. Springer, 1998.

- [35] M. Blum. Rzut monetą przez telefon. w proc. IEEE COMPCOM, strony 133–137, 1982.
- [36] M. Blum i S. Goldwasser. Wydajny probabilistyczny schemat szyfrowania klucza publicznego, który ukrywa wszystkie częściowe informacje. W Advances in Cryptology —Crypto '84, tom 196 notatek z wykładów w informatyce, strony 289–302. Springer, 1985.
- [37] M. Blum i S. Micali. Jak wygenerować silne kryptograficznie sekwencje bitów pseudolosowych. SIAM Journal on Computing, 13 (4): 850–864, 1984.
- [38] D. Boneh. Dwadzieścia lat ataków na kryptosystem RSA. Zawiadomienia Amerykańskiego Towarzystwa Matematycznego, 46(2):203–213, 1999.
- [39] D. Boneh. Uproszczony OAEP dla funkcji RSA i Rabina. W Advances in Cryptology — Crypto 2001, tom 2139 LNCS, strony 275–291. Springer, 2001.
- [40] D. Boneh, A. Joux i PQ Nguyen. Dlaczego podręcznikowe szyfrowanie ElGamal i RSA jest niebezpieczne. W Advances in Cryptology —Asiacrypt 2000, tom 1976 LNCS, strony 30–43. Springer, 2000.
- [41] R. Canetti, O. Goldreich i S. Halevi. Ponowna analiza metodologii losowej wyroczni. Dziennik ACM, 51(4):557–594, 2004.
- [42] L. Carter i MN Wegman. Uniwersalne klasy funkcji skrótu. J. Nauki komputerowe i systemowe, 18 (2): 143–154, 1979.
- [43] D. Chaum, E. van Heijst i B. Pfitzmann. Kryptograficzne mocne, niezaprzeczalne podpisy, bezwarunkowo bezpieczne dla podpisującego. W Advances in Cryptology —Crypto '91, tom 576 LNCS, strony 470–484. Springer, 1992.
- [44] LN Childs. Konkretnie wprowadzenie do wyższej algebra. Teksty licencjackie z matematyki. Springer, wydanie drugie, 2000.
- [45] D. Kotlarz. Standard szyfrowania danych (DES) i jego siła przed atakami. IBM Journal of Research and Development, 38(3):243–250, 1994.
- [46] D. Kotlarz. Małe rozwiązania równań wielomianowych i podatność na RSA o niskim wykładniku. Journal of Cryptology, 10 (4): 233–260, 1997.
- [47] D. Coppersmith, MK Franklin, J. Patarin i MK Reiter. RSA o niskim wykładniku z powiązanymi komunikatami. W Advances in Cryptology —Eurocrypt '96, tom 1070 LNCS, strony 1–9. Springer, 1996.

- [48] J.-S. Coron, Y. Dodis, C. Malinaud i P. Puniya. Ponowna wizyta Merkle-Damgård: Jak skonstruować funkcję skrótu. W *Advances in Cryptology — Crypto 2005*, tom 3621 LNCS, strony 430–448. Springer, 2005.
- [49] J.-S. Coron, M. Joye, D. Naccache i P. Paillier. Nowe ataki na szyfrowanie PKCS #1 v1.5. W *Advances in Cryptology — Eurocrypt 2000*, tom 1807 LNCS, strony 369–381. Springer, 2000.
- [50] R. Cramer i V. Shoup. Projektowanie i analiza praktycznych schematów szyfrowania kluczem publicznym zabezpieczającym przed adaptacyjnym atakiem wybranego tekstu zaszyfrowanego. *SIAM Journal on Computing*, 33 (1): 167–226, 2003.
- [51] R. Crandall i C. Pomerance. *Liczby pierwsze: obliczenia Perspektywiczny*. Springer, wydanie 2, 2005.
- [52] I. Damgard. Bezkolizyjne funkcje skrótu i schematy podpisu klucza publicznego. W *Advances in Cryptology — Eurocrypt '87*, tom 304 LNCS, strony 203–216. Springer, 1988.
- [53] I. Damgard. Zasada projektowania funkcji skrótu. W *Advances in Cryptology — Crypto '89*, tom 435 LNCS, strony 416–427. Springer, 1990.
- [54] JP Degabriele i KG Paterson. O (nie)bezpieczeństwie IPsec w konfiguracjach MAC-then-encrypt. W 17. konferencji ACM. w sprawie bezpieczeństwa komputerów i komunikacji, strony 493–504. Prasa ACM, 2010.
- [55] J. DeLaurentis. Kolejna słabość wspólnego protokołu modułu dla kryptoalgorytmu RSA. *Cryptologia*, 8: 253–259, 1984.
- [56] G. Di Crescenzo, J. Katz, R. Ostrovsky i A. Smith. Efektywne i nieinteraktywne, nieplastyczne zaangażowanie. W *Advances in Cryptology — Eurocrypt 2001*, tom 2045 LNCS, strony 40–59. Springer, 2001.
- [57] M. Dietzfelbinger. Testowanie pierwszości w czasie wielomianowym. Skoczek, 2004.
- [58] W. Diffie i M. Hellman. Nowe kierunki w kryptografii. *Transakcje IEEE dotyczące teorii informacji*, 22 (6): 644–654, 1976.
- [59] W. Diffie i M. Hellman. Wyczerpująca kryptoanaliza standardu szyfrowania danych NBS. *Komputer*, strony 74–84, czerwiec 1977.
- [60] JD Dixon. Asymptotycznie szybka faktoryzacja liczb całkowitych. *Matematyka obliczeniowa*, 36: 255–260, 1981.
- [61] D. Dolev, C. Dwork i M. Naor. Kryptografia nieplastyczna. *SIAM J. Computing*, 30(2):391–437, 2000. Wersja wstępna w STOC '91.

- [62] C. Ellison i B. Schneier. Dziesięć zagrożeń związanych z PKI: czego nie mówi się o infrastrukturze klucza publicznego. *Dziennik bezpieczeństwa komputerowego*, 16 (1): 1–7, 2000.
- [63] S. Even i Y. Mansour. Konstrukcja szyfru z pojedynczej permutacji pseudolosowej. *J. Kryptologia*, 10(3):151–162, 1997.
- [64] H. Feistel. Kryptografia i prywatność komputera. *Scientific American*, 228 (5): 15–23, 1973.
- [65] A. Fiat i A. Shamir. Jak się wykazać: praktyczne rozwiązania problemów z identyfikacją i podpisem. W *Advances in Cryptology —Crypto '86*, tom 263 LNCS, strony 186–194. Springer, 1987.
- [66] R. Fischlin i C.-P. Schnorra. Silniejsze dowody bezpieczeństwa dla bitów RSA i Rabin. *Journal of Cryptology*, 13 (2): 221–244, 2000.
- [67] JB Fraleigh. Pierwszy kurs algebra abstrakcyjnej. Addison Wesley, wydanie 7, 2002.
- [68] E. Fujisaki, T. Okamoto, D. Pointcheval i J. Stern. RSA-OAEP jest bezpieczny przy założeniu RSA. *Journal of Cryptology*, 17 (2): 81–104, 2004.
- [69] SD Galbraith. Matematyka kryptografii klucza publicznego. Cambridge University Press, 2012.
- [70] T. El Gamal. Kryptosystem klucza publicznego i schemat podpisu oparty na logarytmach dyskretnych. *IEEE Trans. Informacje. Teoria*, 31 (4): 469–472, 1985.
- [71] CF Gauss. *Disquisitiones Arithmeticae*. Springer, 1986. (wyd. angielskie cja).
- [72] R. Gennaro, Y. Gertner i J. Katz. Dolne granice wydajności schematów szyfrowania podpisu cyfrowego. W 35. dorocznym sympozjum ACM na temat teorii informatyki, strony 417–425. Prasa ACM, 2003.
- [73] EN Gilbert, FJ MacWilliams i NJA Sloane. Kody wykrywające oszustwo. *Dziennik techniczny Bell Systems*, 53 (3): 405–424, 1974.
- [74] O. Goldreich. Dwie uwagi dotyczące schematu podpisu Goldwasser-Micali-Rivest. W *Advances in Cryptology —Crypto '86*, tom 263 LNCS, strony 104–110. Springer, 1987.
- [75] O. Goldreich. Podstawy kryptografii, tom. 1: Podstawowe narzędzia. Cambridge University Press, 2001.
- [76] O. Goldreich. Podstawy kryptografii, tom. 2: Podstawowe zastosowania. Cambridge University Press, 2004.

- [77] O. Goldreich, S. Goldwasser i S. Micali. O kryptograficznych zastosowaniach funkcji losowych. W *Advances in Cryptology —Crypto '84*, tom 196 LNCS, strony 276–288. Springer, 1985.
- [78] O. Goldreich, S. Goldwasser i S. Micali. Jak konstruować funkcje losowe. *Journal of the ACM*, 33(4):792–807, 1986.
- [79] O. Goldreich i LA Levin. Twardy predykat dla wszystkich funkcji jednokierunkowych. W 21. dorocznym sympozjum ACM na temat teorii informatyki, strony 25–32. Prasa ACM, 1989.
- [80] S. Goldwasser i S. Micali. Szyfrowanie probabilistyczne. *Dziennik Computer i System Sciences*, 28(2):270–299, 1984.
- [81] S. Goldwasser, S. Micali i R. Rivest. Schemat podpisu cyfrowego zabezpieczający przed atakami adaptacyjnymi z użyciem wybranych wiadomości. *SIAM J. Computing*, 17(2):281–308, 1988.
- [82] S. Goldwasser, S. Micali i AC-C. Yao. Silne schematy podpisów. w proc. 15. doroczne sympozjum ACM na temat teorii informatyki, strony 431–439. AKM, 1983.
- [83] D. Hankerson, AJ Menezes i SA Vanstone. *Przewodnik po kryptografii krzywych eliptycznych*. Springer, 2004.
- [84] J. Hastad. Rozwiązywanie równoczesnych równań modułowych niskiego stopnia. *SIAM Journal on Computing*, 17 (2): 336–341, 1988.
- [85] J. Hastad, R. Impagliazzo, L. Levin i M. Luby. Generator pseudolosowy z dowolnej funkcji jednokierunkowej. *SIAM Journal on Computing*, 28 (4): 1364–1396, 1999.
- [86] J. Hastad i M. Nørlund. Bezpieczeństwo wszystkich RSA i dzienników dyskretnych bity. *Dziennik ACM*, 51(2):187–230, 2004.
- [87] M. Hellman. Kryptanalytyczny kompromis w zakresie pamięci i czasu. *IEEE Trans. W-Teoria formacji*, 26(4):401–406, 1980.
- [88] N. Heninger, Z. Durumeric, E. Wustrow i JA Halderman. Wydobywanie dwóch Ps i Qs: Wykrywanie szeroko rozpowszechnionych słabych kluczy w urządzeniach sieciowych. w proc. 21. Sympozjum Bezpieczeństwa USENIX, 2012.
- [89] W Hersteinie. *Algebra abstrakcyjna*. Wiley, wydanie 3, 1996.
- [90] HM Hej. Projektowanie szyfrów sieciowych substytucyjno-permutacyjnych odpornych na kryptoanalizę. Rozprawa doktorska, Queen's University, 1994.
- [91] HM Hej. Samouczek na temat kryptonalu liniowego i różnicowego - dostępny również tak. *Cryptologia*, 26(3):189–221, 2002. <http://www.engr.mun.ca/~howard/Research/Papers/>.

- [92] D. Hofheinz i E. Kiltz. Praktyczne, wybrane, bezpieczne szyfrowanie tekstu zaszyfrowanego z faktoringu. W Advances in Cryptology —Eurocrypt 2009, tom 5479 LNCS, strony 313–332. Springer, 2009.
- [93] R. Impagliazzo i M. Luby. Funkcje jednokierunkowe są niezbędne dla kryptografii oparta na złożoności. W 30. Dorocznym Sympozjum na temat podstaw informatyki, s. 230–235. IEEE, 1989.
- [94] ISO/IEC 9797. Techniki kryptograficzne danych – mechanizm integralności danych wykorzystujący funkcję kontroli kryptograficznej wykorzystującą szyfr blokowy algorytm, 1989.
- [95] T. Iwata i K. Kurosawa. OMAC: Jednoprzyciskowy CBC MAC. W Szybkim Szyfrowaniu oprogramowania —FSE 2003, tom 2887 LNCS, strony 129–153. Springer, 2003.
- [96] A. Joux. Algorytmiczna kryptoanaliza. Chapman i Hall/CRC Press, 2009.
- [97] D. Kahn. The Codebreakers: kompleksowa historia tajnej komunikacji od czasów starożytnych do Internetu. Scribnera, 1996.
- [98] J. Katz. Podpisy cyfrowe. Springer, 2010.
- [99] J. Katz i C.-Y. Koo. O konstruowaniu uniwersalnych jednokierunkowych funkcji skrótu z dowolnych jednokierunkowych funkcji. J. Kryptologia, wystąpić. Dostępne pod adresem <http://eprint.iacr.org/2005/328>.
- [100] J. Katz i M. Yung. Niepodrabialne szyfrowanie i wybrany tekst zaszyfrowany bezpieczne tryby pracy. W szybkim szyfrowaniu oprogramowania —FSE 2000, tom 1978 LNCS, strony 284–299. Springer, 2000.
- [101] J. Katz i M. Yung. Charakterystyka pojęć bezpieczeństwa probabilistycznego szyfrowania klucza prywatnego. Journal of Cryptology, 19 (1): 67–96, 2006.
- [102] C. Kaufman, R. Perlman i M. Speciner. Bezpieczeństwo sieci: prywatne Komunikacja w świecie publicznym. Prentice Hall, wydanie drugie, 2002.
- [103] A. Kerckhoffs. Militarna kryptografia. Journal des Sciences Militaires, IX: 5–38, styczeń 1883. Kopia artykułu jest dostępna pod adresem <http://www.petitcolas.net/fabien/kerckhoffs>.
- [104] A. Kerckhoffs. Militarna kryptografia. Journal des Sciences Militaires, IX: 161–191, luty 1883. Kopia artykułu jest dostępna pod adresem <http://www.petitcolas.net/fabien/kerckhoffs>.
- [105] J. Kilian i P. Rogaway. Jak chronić DES przed kluczem wyczerpującym wyszukiwanie (analiza DESX). Journal of Cryptology, 14 (1): 17–35, 2001.
- [106] L. Knudsen i MJB Robshaw. Towarzysz szyfru blokowego. Springer, 2011.

- [107] LM Kohnfelder. W kierunku praktycznego kryptosystemu klucza publicznego, 1978. Praca licencjacka, MIT.
- [108] H. Krawczyk. Kolejność szyfrowania i uwierzytelniania w celu ochrony komunikacji (lub: Jak bezpieczny jest SSL?). W Advances in Cryptology —Crypto 2001, tom 2139 LNCS, strony 310–331. Springer, 2001.
- [109] H. Krawczyk. Ekstrakcja kryptograficzna i wyprowadzanie klucza: schemat HKDF. W Advances in Cryptology —Crypto 2010, tom 6223 LNCS, strony 631–648. Springer, 2010.
- [110] T. Krovetz i P. Rogaway. Wydajność oprogramowania w trybach uwierzytelnionego szyfrowania. W Fast Software Encryption —FSE 2011, tom 6733 LNCS, strony 306–327. Springer, 2011.
- [111] L. Lamporta. Konstruowanie podpisów cyfrowych z funkcji jednokierunkowej. Raport techniczny CSL-98, SRI International, 1978.
- [112] AK Lenstra, JP Hughes, M. Augier, JW Bos, T. Kleinjung i C. Wachter. Klucze publiczne. W Advances in Cryptology —Crypto 2012, tom 7417 LNCS, strony 626–642. Springer, 2012.
- [113] AK Lenstra i ER Verheul. Wybór rozmiarów kluczy kryptograficznych. Dzień na Cryptology, 14(4):255–293, 2001.
- [114] S. Levy. Krypto: Jak rebelianci Kodeksu pokonali rząd – oszczędzając Prywatność w epoce cyfrowej. Wiking, 2001.
- [115] M. Luby. Pseudolosowość i zastosowania kryptograficzne. Princeton Wydawnictwo Uniwersyteckie, 1996.
- [116] M. Luby i C. Rackoff. Jak konstruować permutacje pseudolosowe z funkcji pseudolosowych. SIAM J. Computing, 17(2):373–386, 1988.
- [117] J. Manger. Wybrany atak szyfrogramem na optymalne asymetryczne wypełnienie szyfrowania RSA (OAEP) zgodnie ze standardem w PKCS #1 v2.0. W Advances in Cryptology —Crypto 2001, tom 2139 LNCS, strony 230–238. Springer, 2001.
- [118] M. Matsui. Metoda kryptoanalizy liniowej dla szyfru DES. W Advances in Cryptology —Eurocrypt '93, tom 765 LNCS, strony 386–397. Springer, 1993.
- [119] A. Maj. Nowe luki w zabezpieczeniach RSA wykorzystujące metody redukcji sieci. Rozprawa doktorska, Uniwersytet w Paderborn, 2003.
- [120] AJ Menezes, PC van Oorschot i SA Vanstone. Podręcznik Kryptografia stosowana. Prasa CRS, 1997.

- [121] RC Merkle. Podpis cyfrowy oparty na konwencjonalnej funkcji szyfrowania. W Advances in Cryptology —Crypto '87, tom 293 LNCS, strony 369–378. Springer, 1988.
- [122] RC Merkle. Certyfikowany podpis cyfrowy. W Advances in Cryptology —Crypto '89, tom 435 LNCS, strony 218–238. Springer, 1990.
- [123] RC Merkle. Jednokierunkowe funkcje skrótu i DES. W Advances in Cryptology —Crypto '89, tom 435 LNCS, strony 428–446. Springer, 1990.
- [124] RC Merkle i M. Hellman. O bezpieczeństwie wielokrotnego szyfrowania. Komunikaty ACM, 24(7):465–467, 1981.
- [125] S. Micali, C. Rackoff i B. Sloan. Pojęcie bezpieczeństwa probabilistycznych kryptosystemów. SIAM J. Computing, 17(2):412–426, 1988.
- [126] E. Miles i E. Viola. Sieci podstawieniowo-permutacyjne, funkcje pseudolosowe i dowody naturalne. W Advances in Cryptology —Crypto 2012, tom 7417 LNCS, strony 68–85. Springer, 2012.
- [127] GL Millera. Hipoteza Riemanna i testy na pierwszość. Journal of Computer and System Sciences, 13 (3): 300–317, 1976.
- [128] M. Naor i M. Yung. Uniwersalne jednokierunkowe funkcje skrótu i ich zastosowania kryptograficzne. w proc. 21. doroczne sympozjum ACM na temat teorii informatyki, strony 33–43. AKM, 1989.
- [129] M. Naor i M. Yung. Kryptosystemy z kluczem publicznym w sposób udowodniony zabezpieczają przed wybranymi atakami zaszyfrowanymi tekstami. W 22. dorocznym sympozjum ACM na temat teorii informatyki, strony 427–437. Prasa ACM, 1990.
- [130] Krajowe Biuro Normalizacyjne. Federalny standard przetwarzania informacji publikacja 81: Tryby pracy DES, 1980.
- [131] Narodowy Instytut Standardów i Technologii. Publikacja federalnej normy przetwarzania informacji 198-1: Kod uwierzytelniania wiadomości z kluczem skrótu (HMAC), lipiec 2008 r.
- [132] Narodowy Instytut Standardów i Technologii. Publikacja federalnych standardów przetwarzania informacji 186-4: Standard podpisu cyfrowego (DSS), lipiec 2013 r.
- [133] VI Nieczajew. Złożoność wyznaczonego algorytmu dla logarytmu dyskretnego. Notatki matematyczne, 55 (2): 165–172, 1994.
- [134] P. Oechslin. Dokonanie szybszego kryptoanalytycznego kompromisu w zakresie czasu i pamięci. W Advances in Cryptology —Crypto 2003, tom 2729 LNCS, strony 617–630. Springer, 2003.

- [135] C. Paar i J. Pelzl. *Zrozumienie kryptografii*. Springer, 2010.
- [136] P. Paillier. Kryptosystemy klucza publicznego oparte na klasach rezydencji o złożonym stopniu. W *Advances in Cryptology —Eurocrypt '99*, tom 1592 LNCS, strony 223–238. Springer, 1999.
- [137] E. Petrank i C. Rackoff. CBC MAC dla źródeł danych w czasie rzeczywistym. *Dzień-
nal of Cryptology*, 13(3):315–338, 2000.
- [138] S. Pohlig i M. Hellman. Ulepszony algorytm obliczania logarytmów na podstawie GF
(p) i jego znaczenie kryptograficzne. *IEEE Trans.
Teoria informacji*, 24(1):106–110, 1978.
- [139] JM Pollard. Twierdzenia faktoryzacji i testowania pierwszości. *Proc.
Towarzystwo Filozoficzne w Cambridge*, 76: 521–528, 1974.
- [140] JM Pollard. Metoda Monte Carlo faktoryzacji. *BIT Matematyka numeryczna*, 15 (3):
331–334, 1975.
- [141] JM Pollard. Metody Monte Carlo do obliczania indeksów (mod p).
Matematyka obliczeniowa, 32 (143): 918–924, 1978.
- [142] C. Pomerance. Algorytm faktoryzacji przez sito kwadratowe. W *Advances in
Cryptology —Eurocrypt '84*, tom 209 LNCS, strony 169–182.
Springera, 1985.
- [143] B. Preneel, R. Govaerts i J. Vandewalle. Funkcje skrótu oparte na szyfrach blokowych:
podejście syntetyczne. W *Advances in Cryptology —Crypto '93*, tom 773 LNCS,
strony 368–378. Springer, 1994.
- [144] MO Rabin. Cyfrowe podpisy. W RA Demillo, DP Dobkin, AK
Jones i RJ Lipton, redaktorzy, *Foundations of Security Computation*, strony 155–168.
Prasa akademicka, 1978.
- [145] MO Rabin. Cyfrowe podpisy równie trudne do rozwiązania jak faktoryzacja. *Tech-
Raport finansowy TR-212*, MIT/LCS, 1979.
- [146] MO Rabin. Algorytm probabilistyczny do badania pierwszości. *Dziennik
Teoria liczb*, 12 (1): 128–138, 1980.
- [147] C. Rackoff i DR Simon. Nieinteraktywny dowód wiedzy z wiedzą zerową i atak
wybranym szyfrogramem. W *Advances in Cryptology —Crypto '91*, tom 576 LNCS,
strony 433–444. Springer, 1992.
- [148] R. Rivest, A. Shamir i L. Adleman. Metoda uzyskiwania podpisów cyfrowych i
kryptosystemów klucza publicznego. *Komunikaty ACM*, 21(2):120–126, 1978.
- [149] P. Rogaway. Bezpieczeństwo DESX. *CryptoBytes RSA Laboratories*, lato 1996.

- [150] P. Rogaway i T. Shrimpton. Podstawy kryptograficznej funkcji skrótu: definicje, implikacje i separacje dotyczące odporności na obraz wstępny, odporność na drugi obraz wstępny i odporność na kolizje. W Szybkim Oprogramowaniu Szyfrowanie – FSE 2004, tom 3017 LNCS, strony 371–388. Skoczek, 2004.
- [151] J. Rompel. Funkcje jednokierunkowe są konieczne i wystarczające dla bezpieczeństwa podpisów. w proc. 22. doroczne sympozjum ACM na temat teorii informatyki, strony 387–394. AKM, 1990.
- [152] C.-P. Schnorra. Efektywna identyfikacja i podpisy dla kart inteligentnych. W Advances in Cryptology —Crypto '89, tom 435 LNCS, strony 239–252. Springer, 1990.
- [153] D. Shanks. Numer klasy, teoria faktoryzacji i rodzaje. w proc. Sympozja w czystej matematyce 20, strony 415–440. Amerykańskie Towarzystwo Matematyczne, 1971.
- [154] CE Shannon. Teoria komunikacji systemów tajnych. Systemy dzwonkowe Dziennik techniczny, 28 (4): 656–715, 1949.
- [155] V. Shoup. Dolne granice logarytmów dyskretnych i problemy pokrewne. W Advances in Cryptology —Eurocrypt '97, tom 1233 LNCS, strony 256–266. Springer, 1997.
- [156] V. Shoup. Dlaczego bezpieczeństwo wybranego tekstu zaszyfrowanego ma znaczenie. Raport cal RZ 3076, IBM Zurich, listopad 1998. Dostępne pod adresem <http://shoup.net/papers/expo.pdf>. Technika-
- [157] V. Shoup. Propozycja standardu ISO dotyczącego szyfrowania klucza publicznego, 2001. Dostępne pod adresem <http://eprint.iacr.org/201/112>.
- [158] V. Shoup. Ponownie rozważono OAEP. Journal of Cryptology, 15(4):223–249, 2002.
- [159] V. Shoup. Obliczeniowe wprowadzenie do teorii liczb i Alge-bra. Cambridge University Press, wydanie 2, 2009. Dostępne także pod adresem: <http://www.shopup.net/ntb>.
- [160] JH Silverman i J. Tate. Racjonalne punkty na krzywych eliptycznych. Teksty licencjackie z matematyki. Springer, 1994.
- [161] G.Simmons. „Słaby” protokół prywatności wykorzystujący algorytm kryptograficzny RSA rytm. Kryptologia, 7: 180–182, 1983.
- [162] G.Simmons. Badanie uwierzytelniania informacji. U G. Simmonsa, redaktor, Współczesna kryptologia: nauka o integralności informacji, strony 379–419. Prasa IEEE, 1992.

- [163] S. Singh. Księga kodów: nauka o tajemnicy od starożytnego Egiptu do kryptografii kwantowej. Książki kotwiczne, 2000.
- [164] R. Solovay i V. Strassen. Szybki test Monte-Carlo na pierwszość. SYJAM Journal on Computing, 6 (1): 84–85, 1977.
- [165] W. Stallings. Podstawy bezpieczeństwa sieci: aplikacje i standardy. Prentice Hall, wydanie 5, 2013.
- [166] DR Stinson. Uniwersalne kody mieszające i uwierzytelniające. Projekty, kody i kryptografia, 4 (4): 369–380, 1994.
- [167] DR Stinson. Kryptografia: teoria i praktyka . Chapman & Hall/CRC Press, wydanie 1, 1995.
- [168] DR Stinson. Kryptografia: teoria i praktyka . Chapman & Hall/CRC Press, wydanie 3, 2005.
- [169] W. Trappe i L. Washington. Wprowadzenie do kryptografii z teorią kodowania. Prentice Hall, wydanie drugie, 2005.
- [170] PC van Oorschot i MJ Wiener. Równoległe wyszukiwanie kolizji z aplikacjami do analizy krypt. Journal of Cryptology, 12 (1): 1–28, 1999.
- [171] S. Vaudenay. Luki w zabezpieczeniach spowodowane przez dopełnienie CBC —aplikacje do SSL, IPSEC, WTLS, W Advances in Cryptology —Eurocrypt 2002, tom 2332 LNCS, strony 534–546. Springer, 2002.
- [172] GS Vernama. Systemy telegraficzne drukujące szyfry do tajnej komunikacji przewodowej i radiotelegraficznej. Journal of American Institute for Electrical Engineers, 55: 109–115, 1926.
- [173] SS Wagstaff, Jr. Kryptoanaliza szyfrów teoretycznych liczb. Chapman i Hall/ CRC Press, 2003.
- [174] X. Wang, YL Yin i H. Yu. Znajdowanie kolizji w pełnym SHA-1. W Advances in Cryptology —Crypto 2005, tom 3621 LNCS, strony 17–36. Springer, 2005.
- [175] X. Wang i H. Yu. Jak złamać MD5 i inne funkcje skrótu. W Advances in Cryptology —Eurocrypt 2005, tom 3494 LNCS, strony 19–35. Springer, 2005.
- [176] L. Waszyngton. Krzywe eliptyczne: teoria liczb i kryptografia. Chapman i Hall/CRC Press, 2003.
- [177] MN Wegman i L. Carter. Nowe funkcje skrótu i ich zastosowanie w uwierzytelnianiu i ustawianiu równości. J. Computer and System Sciences, 22(3):265–279, 1981.

- [178] ANSI X9.9. Amerykański krajowy standard dotyczący protokołów instytucji finansowych uwierzytelnianie mędrcza (hurt), 1981.
- [179] AC-C. Yao. Teoria i zastosowania funkcji zapadni. W 23 Coroczne sympozjum na temat podstaw informatyki, strony 80–91. IEEE, 1982.
- [180] G. Yuval. Jak oszukać Rabina. Cryptologia, 3: 187–189, 1979.

Informatyka/matematyka

Kryptografia jest wszechobecna i odgrywa kluczową rolę w zapewnianiu poufności i integralności danych, a także w szerzej rozumianym zabezpieczaniu systemów komputerowych. Wprowadzenie do współczesnej kryptografii zapewnia rygorystyczne, ale przystępne podejście do tego fascynującego tematu.

Autorzy przedstawiają podstawowe zasady współczesnej kryptografii, kładąc nacisk na formalne definicje, jasne założenia i rygorystyczne dowody bezpieczeństwa. Książka rozpoczyna się od skupienia się na kryptografii klucza prywatnego, w tym obszernego omówienia szyfrowania klucza prywatnego, kodów uwierzytelniających wiadomości i funkcji skrótu. Autorzy przedstawiają także zasady projektowania powszechnie używanych szyfrów strumieniowych i szyfrów blokowych, w tym RC4, DES i AES, a także dostarczają możliwych do udowodnienia konstrukcji szyfrów strumieniowych i szyfrów blokowych z prymitywów niższego poziomu. Druga połowa książki obejmuje kryptografię klucza publicznego, zaczynając od samodzielnego wprowadzenia do teorii liczb potrzebnej do zrozumienia kryptosystemów RSA, Diffiego-Hellmana i El Gamala (i innych), po którym następuje dokładne omówienie kilku standardowych schematów szyfrowania kluczem publicznym i podpisu cyfrowego.

Integrując bardziej praktyczną perspektywę bez poświęcania dyscypliny, to szeroko oczekiwane drugie wydanie oferuje ulepszone podejście do zagadnień

- Szyfry strumieniowe i szyfry blokowe, łącznie z trybami działania i zasady projektowania
- Uwierzytelnione szyfrowanie i bezpieczne sesje komunikacyjne
- Funkcje skrótu, w tym aplikacje i projekty funkcji skrótu zasady
- Ataki na źle zaimplementowaną kryptografię, w tym ataki na szyfrowanie łańcuchowe CBC, ataki typu padding-oracle i ataki czasowe
- Model Random-Oracle i jego zastosowanie w kilku standardowych, szeroko stosowanych schematach szyfrowania i podpisu z kluczem publicznym
- Kryptografia krzywej eliptycznej i powiązane standardy, takie jak DSA/ECDSA i DHIES/ECIES

Zawierające aktualne ćwiczenia i praktyczne przykłady, Wprowadzenie do współczesnej kryptografii, wydanie drugie, może służyć jako podręcznik do kursów na poziomie licencjackim lub magisterskim z kryptografią, cenne źródło informacji dla badaczy i praktyków lub ogólnie wprowadzenie odpowiednie do samodzielnnej nauki.



CRC Press
Taylor & Francis Group
an informa business
www.crcpress.com

6000 Broken Sound Parkway, NW
Suite 300, Boca Raton, FL 33487
711 Third Avenue
New York, NY 10017
2 Park Square, Milton Park
Abingdon, Oxon OX14 4RN, UK

K16475
ISBN : 978-1-4665-7026-9
9 781466 570269

www.crcpress.com