Algorytmy kryptograficzne

Przewodnik po algorytmach w blockchain, kryptografii kwantowej, protokołach o wiedzy zerowej oraz szyfrowaniu homomorficznym

Massimo Bertaccini

Helion

Tytuł oryginału: Cryptography Algorithms: A guide to algorithms in blockchain, quantum cryptography, zero-knowledge protocols, and homomorphic encryption

Tłumaczenie: Robert Górczyński

ISBN: 978-83-289-0012-7

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/algkry
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
https://ftp.helion.pl/przyklady/algkry.zip

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

Spis treści

CZĘŚĆ 3. Protokoły i algorytmy nowej kryptografii

CZĘŚĆ 4. Szyfrowanie homomorficzne i silnik CSE

Rozdział 3

Szyfrowanie asymetryczne

Szyfrowanie asymetryczne oznacza zastosowanie różnych par kluczy do szyfrowania i deszyfrowania wiadomości. Synonimem takiego szyfrowania jest szyfrowanie z użyciem klucza prywatnego i publicznego, choć istnieją pewne różnice między szyfrowaniem asymetrycznym i szyfrowaniem z użyciem klucza prywatnego i publicznego, które zostanie dokładniej omówione w dalszej części rozdziału. Rozpocznę od odrobiny historii tej rewolucyjnej metody szyfrowania i deszyfrowania, później przejdę do przedstawienia różnych typów algorytmów szyfrowania asymetrycznego i wyjaśnienia, jak pomagają one zabezpieczać nasze karty kredytowe, tożsamość i dane.

W tym rozdziale zostaną poruszone następujące zagadnienia:

- szyfrowanie asymetryczne i szyfrowanie z użyciem klucza prywatnego i publicznego;
- protokół przekazywania klucza Diffiego-Hellmana i związany z nim problem pośrednika;
- algorytm RSA i interesujące zagrożenia międzynarodowe;
- wprowadzenie do konwencjonalnych i niekonwencjonalnych ataków na RSA;
- **Pretty Good Privacy** (**PGP**);
- algorytm ElGamal i jego potencjalne luki w zabezpieczeniach.

Zaczynamy!

Wprowadzenie do szyfrowania asymetrycznego

Najważniejszą funkcją szyfrowania z użyciem klucza prywatnego i publicznego jest przekazywanie klucza między dwiema stronami oraz zapewnienie bezpieczeństwa informacji podczas transakcji.

Aby w pełni zrozumieć szyfrowanie asymetryczne, trzeba poznać jego historię. Okazuje się ono szczególnie ważne w naszym życiu codziennym. Ta gałąź kryptografii odpowiada za zabezpieczanie informacji finansowych, takich jak karty kredytowe i bankowość internetowa, służy do generowania używanych przez nas haseł oraz ogólnie do bezpiecznego przekazywania danych wrażliwych i zapewnienia prywatności.

Warto dowiedzieć się nieco więcej na temat historii tej fascynującej gałęzi kryptografii.

Historia kryptografii asymetrycznej rozpoczyna się w późnych latach siedemdziesiątych ubiegłego wieku, a rozwój tej gałęzi przypada na lata osiemdziesiąte, gdy rozwój gospodarki cyfrowej i internetu doprowadził do pojawienia się komputerów w domach. Późne lata siedemdziesiąte ubiegłego wieku to czas, w którym Steve Jobs założył firmę Apple Inc., a między USA i ZSRR trwała zimna wojna. To był również czas wzrostu gospodarczego w wielu krajach Zachodu, takich jak Włochy, Francja i Niemcy. Wówczas pojawił się internet. Dwa przeciwne bloki, Zachód i Wschód, po jednej stronie USA razem z sojusznikami, a po drugiej blok sowiecki, doprowadziły do powstania siatek szpiegowskich, dla których ważnym miejscem było podzielone miasto Berlin. W tym czasie przekazywanie kluczy stosowanych w kryptografii symetrycznej osiągnęło poziom, na którym amerykańska organizacja rządowa **COMSEC**, odpowiedzialna za przekazywanie kluczy kryptograficznych, przekazywała ich tony każdego dnia. To była problematyczna sytuacja prowadząca do punktu, w którym system mógł się załamać. Dla przykładu w przypadku algorytmu DES w latach siedemdziesiątych banki przekazywały klucze za pośrednictwem kurierów dostarczających je osobiście. **Narodowa Agencja Bezpieczeństwa** w USA zmagała się z problemem dostarczania kluczy pomimo dostępu do najlepszych zasobów komputerowych na świecie. Problem związany z przekazywaniem kluczy wydawał się nie do pokonania, nawet dla ogromnych korporacji utworzonych w celu rozwiązywania największych problemów świata. Przykładem takiej korporacji jest RAND — potężna instytucja utworzona w celu radzenia sobie z przyszłymi problemami i pomagająca w zapobieganiu krytycznym sytuacjom. Uważam, że sytuacja krytyczna jest niekiedy lepszym sposobem na rozwiązanie problemu niż unikanie zmierzenia się z problemem. Czasami problem można rozwiązać na różne sposoby. W przypadku szyfrowania asymetrycznego żadna kwota pieniędzy płynących z rządu, żadne superkomputery o ogromnej mocy obliczeniowej ani żadne potężne umysły nie były w stanie rozwiązać problemu, który na pierwszy rzut oka wydawał się dość prosty do rozwiązania.

Teraz już wiesz, że najważniejszym problemem rozwiązywanym przez szyfrowanie asymetryczne jest przekazywanie klucza. (W rzeczywistości zobaczysz, że w algorytmie RSA ten problem przekłada się na bezpośrednie przekazywanie wiadomości). Mogę więc przedstawić pionierów zaangażowanych w historię tej wyjątkowo intrygującej gałęzi kryptografii.

Pionierzy

Osoby zajmujące się kryptografią często są postrzegane jako dziwne, czasami są introwertykami, a nierzadko wręcz przeciwnie. Tak jest w przypadku *Whitfielda Diffiego*, niezależnego wolnomyśliciela, nie zatrudnionego przez żadną organizację rządową ani dużą korporację. Po raz pierwszy spotkałem go w 2016 roku podczas konwencji w San Francisco, podczas której omawiał kryptografię razem ze sławnymi kolegami: *Martinem Hellmanem* i *Ronaldem Rivesem*. W pamięci najbardziej utkwiły mi jego wysoki wzrost, elegancki biały strój oraz długie siwe włosy i broda. Taki wiecznie młody facet wciąż tkwiący w latach sześćdziesiątych, którego rówieśnikiem mógłby być makler na giełdzie papierów wartościowych albo mnich w Indiach. Jest jednym z prekursorów nowoczesnej kryptografii, a jego nazwisko na zawsze zapisało się w historii szyfrowania z użyciem klucza prywatnego i publicznego. Diffie urodził się w 1944 roku i ukończył MIT w 1965 roku. Po studiach znalazł zatrudnienie w branży kryptograficznej, a później stał się jednym z najbardziej niezależnych kryptografów lat siedemdziesiątych. Zyskał miano **cyberpunka**, w hołdzie dla *nowej fali* ruchu sci-fi lat sześćdziesiątych i siedemdziesiątych, w której cybernetyka, sztuczna inteligencja i kultura hakerów połączyły się w futurystyczną scenerię, koncentrującą się na *połączeniu zwykłego życia i wysokiej technologii*.

W latach sześćdziesiątych Departament Obrony USA rozpoczął finansowanie nowego zaawansowanego programu badawczego z dziedziny komunikacji pod nazwą **Defense Advanced Research Projects Agency** (**DARPA** lub **ARPA**). Głównym celem projektu ARPA było połączenie komputerów wojskowych w celu zapewnienia większego poziomu bezpieczeństwa w telekomunikacji. Dzięki temu projektowi komunikacja miała być możliwa nawet w przypadku ataku nuklearnego. Utworzona sieć pozwoliła również na przekazywanie komunikacji między naukowcami, a także na wykonywanie obliczeń, aby w ten sposób wykorzystać wolne moce obliczeniowe komputerów w sieci. Działanie sieci **ARPANET** oficjalnie rozpoczęło się w 1969 roku. Na początku połączone były jedynie cztery ośrodki. Jednak ta liczba rosła na tyle szybko, że w 1982 roku powstał internet. Pod koniec lat dziewięćdziesiątych coraz więcej zwykłych użytkowników miało połączenie z internetem, a tym samym liczba internautów zaczęła gwałtownie rosnąć.

Diffie uważał, że rozwój sieci ARPANET sprawi, iż pewnego dnia każdy będzie miał komputer i będzie mógł komunikować się z innymi użytkownikami, np. za pomocą poczty elektronicznej. Wyobrażał sobie również świat, w którym produkty są sprzedawane przez internet, a fizyczne pieniądze zostały zastąpione kartami kredytowymi. Jego wnikliwe rozważania dotyczące prywatności i bezpieczeństwa danych doprowadziły do obsesji związanej z problemem, jak prowadzić komunikację z innymi użytkownikami, nawet nie wiedząc, kim oni są. Co więcej, szyfrowanie wiadomości i dokumentów odbywa się często podczas przekazywania niezwykle cennych informacji. Szyfrowanie

danych zaczęło być wykorzystywane przez szerszą grupę osób w celu ukrywania informacji i dzielenia się tajemnicami. W tym czasie stosowanie kryptografii zaczęło stawać się powszechne nie tylko w organizacjach wojskowych, rządowych i akademickich.

Największy problem do rozwiązania przedstawiał się następująco: gdy dwie zupełnie obce sobie osoby spotkają się w internecie, jak można szyfrować i deszyfrować współdzielony dokument bez konieczności wymieniania się jakimikolwiek informacjami dodatkowymi poza wspomnianym dokumentem, który jest zaszyfrowany za pomocą parametrów matematycznych? To w zasadzie sprowadza się do problemu związanego z przekazaniem klucza.

Pewnego dnia w 1974 roku Diffie odwiedził laboratorium IBM im. *Thomasa Watsona*, gdzie miał wygłosić wykład. Tematem były różne strategie ataków na operację przekazywania klucza, przy czym zgromadzeni słuchacze byli bardzo sceptycznie nastawieni do zaproponowanego rozwiązania. Tylko jedna osoba podzieliła wizję Diffiego — starszy kryptograf z firmy IBM, który wspomniał, że laboratorium odwiedził niedawno profesor Uniwersytetu Stanforda i podobnie zapatrywał się na przedstawiony problem. Owym profesorem był Martin Hellman.

Diffie był tak entuzjastycznie nastawiony do projektu, że jeszcze tego samego wieczoru udał się na drugi koniec Ameryki, do Palo Alto w Kalifornii, aby odwiedzić profesora Hellmana. Współpraca między nimi na polu kryptografii doprowadziła do powstania najpiękniejszego algorytmu w tej dziedzinie: **protokołu przekazania klucza Diffiego-Hellmana**.

W następnym podrozdziale przedstawię analizę tego pionierskiego algorytmu.

Algorytm Diffiego-Hellmana

Aby zrozumieć algorytm **Diffiego-Hellmana** (**D-H**), można posłużyć się tzw. **eksperymentami myślowymi** lub **mentalną reprezentacją** teorii, które często stosował Einstein.

Eksperyment myślowy to hipotetyczny scenariusz, w którym mentalnie przenosisz się do bardziej rzeczywistej sytuacji niż jedynie w czysto teoretyczny sposób zastanawiając się nad problemem. Na przykład Einstein używał bardzo popularnego eksperymentu myślowego do wyjaśnienia teorii względności. Posługiwał się metaforą poruszającego się pociągu obserwowanego przez osoby znajdujące się w różnych miejscach, zarówno w pociągu, jak i poza nim.

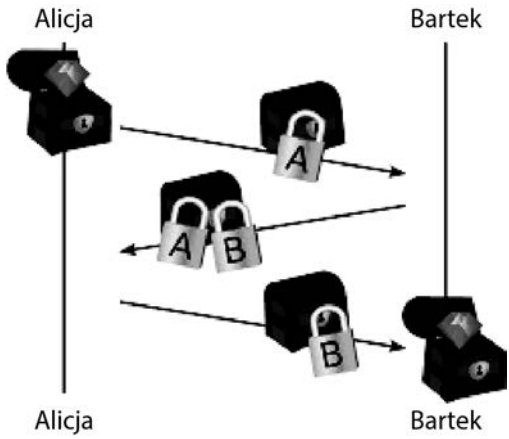
Takie podejście mentalne będę często wykorzystywał w tej książce.

Wyobraźmy sobie dwie osoby, Alicję i Bartka, które chcą wymienić się wiadomościami (papierowymi), ale poczta główna w ich mieście kontroluje zawartość wszystkich listów. Dlatego Alicja i Bob posługują się wieloma różnymi metodami w celu bezpiecznego przekazania wiadomości, chroniąc się przed ciekawskimi. Przykładem może być umieszczenie klucza przez Alicję w zamkniętym metalowym pudełku i przekazanie go Bartkowi. Jednak Bartek nie ma klucza do tego pudełka, więc Alicja i Bartek muszą najpierw się gdzieś spotkać, aby Alicja mogła przekazać Bartkowi klucz. Tym samym wracamy do problemu związanego z przekazaniem klucza.

Mimo podejmowania naprawdę wielu prób wydawało się niemożliwe opracowanie logicznego rozwiązania dla wspomnianego problemu. I wtedy pewnego dnia Diffie przy wsparciu Hellmana znalazł rozwiązanie, o którym Hellman później powiedział: *„Głupi ma szczęście!”*.

Oto mentalna reprezentacja tego, w jaki sposób Alicja i Bartek mogą wymienić się kluczami:

- **Krok 1.** Alicja umieszcza tajną wiadomość w metalowym pudełku zamkniętym na kłódkę i wysyła je Bartkowi, ale zachowuje klucz. Pamiętaj, że Alicja zamknęła, pudełko używając klucza, którego nie przekazała Bartkowi.
- **Krok 2.** Bartek zakłada drugą kłódkę, używając swojego klucza prywatnego, i odsyła pudełko do Alicji. Tak więc po otrzymaniu pudełka od Alicji Bartek nie może go otworzyć. Ograniczył się do założenia swojej kłódki.
- **Krok 3.** Gdy Alicja otrzymuje pudełko z powrotem od Bartka, może zdjąć założoną przez siebie kłódkę, ponieważ pudełko jest zabezpieczone kłódką Bartka (rysunek 3.1). Zauważ, że pudełko zawsze ma założoną jakąś kłódkę. W tym momencie pudełko jest zabezpieczone jedynie kłódką Bartka.



Rysunek 3.1. Algorytm Diffiego-Hellmana na przykładzie wymiany kluczy między Alicją i Bartkiem

- **Krok 4.** Gdy Bartek otrzymuje pudełko po raz drugi, może je otworzyć, ponieważ jedyna założona kłódka jest jego i ma do niej klucz. W efekcie Bartek może odczytać wiadomość umieszczoną przez Alicję w pudełku.

Z opisanych kroków wynika, że Alicja i Bartek nigdy się nie spotkali, aby przekazać sobie klucze do kłódek. Zauważ, że w omawianym przykładzie pudełko było przesyłane dwukrotnie między Alicją i Bartkiem, podczas gdy w algorytmie tak się nie dzieje.

Wprawdzie zamieszczone tutaj wyjaśnienie i schemat nie odpowiadają ściśle sposobowi działania algorytmu, ale dostarczają praktyczne rozwiązanie problemu wydawałoby się niemożliwego do rozwiązania: jak bezpiecznie przekazać klucz.

Teraz te wyjaśnienia praktyczne przekształcę na postać reprezentacji logiczno-matematycznej.

Przede wszystkim powrócę do stosowania arytmetyki modularnej i jednocześnie wykorzystam określone właściwości operacji przeprowadzanych w ciałach skończonych.

Logarytm dyskretny

Matematykę kryjącą się za kryptografią spróbuję wyjaśnić bez używania zbyt wielu notacji, ponieważ nie chcę ani Cię przytłoczyć trudnymi zagadnieniami, ani przeładować książki matematycznymi wywodami.

Gdy mówimy o ciele skończonym, mamy na myśli skończoną grupę (`n`) liczb całkowitych (`Z`) znajdujących się na okręgu (`Zn`). Ta grupa liczb podlega tym samym prawom matematycznym, takim jak operacje na standardowych liczbach całkowitych. Skoro pracujemy z ciałem skończonym, nazywanym tutaj (`modulo n`), konieczne jest rozważenie wybranych ważnych kwestii związanych z arytmetyką modularną. Jak wyjaśniłem w poprzednim rozdziale, operowanie na *modulus* oznacza przejście z powrotem do pierwszej liczby za każdym razem po dotarciu do końca zbioru. To przypomina trochę zegar, gdy po dotarciu wskazówki do godziny 12 następną godziną jest ponownie 1.

Należy pamiętać, że w ciele skończonym istnieje *cykl liczbowy*, w którym liczby i wyniki operacji pojawiają się ponownie. Na przykład jeśli mamy zbiór składający się z siedmiu liczb całkowitych, `{0, 1, 2, 3, 4, 5, 6}`, często przedstawiany w postaci skrótu (`Z7`), mamy wszystkie operacje przeprowadzone wewnątrz tego ciała skończonego, aż do przejścia z powrotem na początek zbioru.

Spójrz na krótki przykład operacji dodawania i mnożenia w ciele skończonym, (`Z7, +, x`). Skoro wszystkie operacje będą działały, (`modulo 7`), konieczne jest określenie, że nastąpi powrót do `0` za każdym razem, gdy operacja przekroczy liczbę `7`:

```
1 x 1 ≡ 1 (mod 7)
2 x 4 ≡ 1 (mod 7)
3 + 5 ≡ 1 (mod 7)
3 x 5 ≡ 1 (mod 7)
```

Wykorzystam teraz tę modularność i pokażę, że notacja = jest odpowiednikiem ≡, a to jest przykład matematyki ze szkoły podstawowej. Tutaj `2 + 2 = 4` nie działa prawidłowo, jeśli rozważymy na przykład ciało skończone `modulo 3`.

```
2 + 2 ≡ 1 (mod 3)
```

Z lekcji matematyki w szkole średniej zapewne pamiętasz, że logarytm `[log a(z)]` to funkcja o podstawie `(a)`. Konieczne jest więc obliczenie wykładnika pozwalającego `(a)` uzyskać liczbę `(z)`. Na przykład jeżeli `a = 10` i `z = 100`, to logarytm wynosi `2`. Można więc powiedzieć, że *logarytm o podstawie 10 dla liczby 100 wynosi 2*. Jeżeli do obliczenia logarytmu zostanie użyty program Mathematica, wówczas jest stosowana inna notacja: `Log[10, 100]= 2`.

W przypadku matematyki dyskretnej sytuacja zaczyna się bardziej komplikować, więc zamiast zwykłego logarytmu używamy logarytmu dyskretnego.

Załóżmy, że mamy do rozwiązania następujące równanie:

```
a^[x] ≡ b (mod p)
```

To będzie bardzo trudny problem do rozwiązania, nawet jeśli znamy wartości `(a)` i `(b)`, ponieważ nie ma efektywnego logarytmu umożliwiającego rozwiązanie logarytmu dyskretnego, czyli `[x]`.

Ważna uwaga

Użyłem nawiasu kwadratowego do określenia, że `[x]` to sekret. Formalnie rzecz biorąc, `[x]` to potęga dyskretna, ale problemy związane z szukaniem logarytmu dyskretnego i potęgi dyskretnej są pod względem obliczeniowym równie skomplikowane.

Zagłębimy się teraz nieco bardziej w problem, aby wyjaśnić dynamikę tej operacji. Przeanalizujmy obliczenie następującego równania:

```
2^4 ≡ (x) (mod 13)
```

Zaczynamy od obliczenia `2^4 = 16`, następnie przeprowadzana jest operacja `16:13 = 1`, a reszta z dzielenia wynosi `3`, więc `x = 3`.

Logarytm dyskretny przedstawia operację odwrotną:

```
2^[y] ≡ (x) (mod 13)
```

W takim przypadku obliczamy `[y]`, wiedząc, że podstawa wynosi `2`. Istnieje możliwość pokazania, że mamy nieskończenie wiele rozwiązań dla `[y]`, które w wyniku dają `(x)`.

Powracając do wcześniejszego przykładu, mamy:

```
2^[y] ≡ 3 (mod 13) for [y]
```

Jednym z rozwiązań jest `y = 4`, ale to nie jest jedyne rozwiązanie.

Wynik `3` będzie spełniony także dla wszystkich liczb całkowitych, `(n)`, przedstawionego tutaj równania.

```
[y] = [y + (p-1)*n]
```

Udowodnijmy je dla `n = 1`:

```
2^[4+(13-1)*1] ≡ 2^16 (mod 13)
2^16 ≡ 3 (mod 13)
```

Prawdziwe jest również dla `n = 2`:

```
2^[4 +(13-1)*2] ≡ 2^28 (mod 13)
2^28 ≡ 3 (mod 13)
```

I tak dalej.

Dlatego równanie ma nieskończoną liczbę rozwiązań dla wszystkich liczb całkowitych, tzn. `(n ≥ 0)`:

```
[y] ≡ 2^[4 + 12 n] (mod 13)
```

Nie istnieje jeszcze metoda umożliwiająca rozwiązanie logarytmu dyskretnego w czasie wielomianowym. Dlatego w matematyce, podobnie jak w kryptografii, ten problem jest uznawany za niezwykle trudny do pokonania, nawet jeśli atakujący dysponuje ogromną mocą obliczeniową.

Konieczne jest również wprowadzenie definicji i notacji generatora, `(g)`, czyli określonej liczby, gdy mówimy, że `(g)` generuje całą grupę, `(Zp)`. Jeżeli `(p)` to liczba pierwsza, wówczas `(g)` może przyjąć dowolną wartość z przedziału od `1` do `p-1`.

Wyjaśnienie algorytmu D-H

D-H nie jest dokładnie algorytmem szyfrowania asymetrycznego, ale może być poprawnie zdefiniowany jako algorytm klucza prywatnego i publicznego. Różnica między szyfrowaniem asymetrycznym oraz kluczem prywatnym i publicznym jest nie tylko formalna, ale dość poważna. Lepiej zrozumiesz ją w dalszej części rozdziału, a dokładnie w podrozdziale poświęconym *RSA*, gdzie przedstawię algorytm czysto asymetryczny. Natomiast D-H służy do pobrania klucza współdzielonego, który jest wykorzystywany do symetrycznego szyfrowania wiadomości, `[M]`.

Szyfrowanie przeprowadzane za pomocą algorytmu symetrycznego zostaje połączone z opartą na protokole D-H operacją przekazania klucza współdzielonego w celu wygenerowania szyfrogramu, C.

```
Symmetric-Algorithm E((D-H[k]), M) = C
```

Innymi słowy algorytm D-H zostaje użyty do wygenerowania współdzielonego tajnego klucza, [k], a następnie za pomocą AES lub innego algorytmu szyfrowania symetrycznego odbywa się zaszyfrowanie wiadomości, [M].

Algorytm D-H nie szyfruje bezpośrednio wiadomości, pozwala jedynie określić klucz współdzielony między dwiema stronami. Jest to punkt o znaczenie krytycznym, o czym się przekonasz w następnym akapicie.

Jednak podczas pracy z matematyką dyskretną i stosowania problemu logarytmu dyskretnego w celu ochrony klucza przed atakującymi podczas jego współdzielenia Diffie i Hellman zaimplementowali jeden z najbardziej niezawodnych i najsławniejszych algorytmów w historii kryptografii.

Zobacz, jak działa algorytm D-H:

- **Krok 1.** Alicja i Bartek zaczynają od uzgodnienia parametrów: generator (g) na okręgu (Zp) i liczba pierwsza p (mod p).
- **Krok 2.** Alicja wybiera tajną liczbę [a], a Bartek [b]. Alicja oblicza A ≡ g^a (mod p), natomiast obliczenia Bartka to B ≡ g^b (mod p).
- **Krok 3.** Alicja przekazuje parametr (A) Bartkowi, który z kolei przekazuje jej parametr (B).
- **Krok 4.** Alicja oblicza ka ≡ B^a (mod p), podczas gdy Bartek oblicza kb ≡ A^b (mod p).

  [ka = kb] to będzie tajna wartość, czyli klucz współdzielony przez Alicję i Bartka.

Oto przykład liczbowy oparty na tym algorytmie:

- **Krok 1.** Alicja i Bartek uzgadniają użyte parametry, czyli w omawianym przykładzie g = 7 i (mod 11).
- **Krok 2.** Alicja wybiera tajną wartość (3), a Bartek (6).

  W przypadku Alicji trzeba obliczyć 7^3 (mod 11) ≡ 2, natomiast Bartek musi obliczyć 7^6 (mod 11) ≡ 4.
- **Krok 3.** Alicja przekazuje (2) Bartkowi, który przekazuje jej (4).
- **Krok 4.** Alicja przeprowadza operację 4^3 (mod 11) ≡ 9, Bartek zaś oblicza 2^6 (mod 11) ≡ 9.

W tym przypadku liczba [9] to tajny klucz [k] współdzielony przez Alicję i Bartka.

Analiza algorytmu

W *kroku 2.* algorytmu Alicja przeprowadza obliczenie `A ≡ g^a (mod p)`, natomiast Bartek oblicza `B ≡ g^b (mod p)`. Alicja i Bartek wymienili się wartościami `(A)` i `(B)`, parametrami publicznymi funkcji jednokierunkowej. Nazwa **funkcji jednokierunkowej** wzięła się stąd, że jest niemożliwe, `(-x%)`, wygenerowanie na podstawie parametru publicznego,`(A)` klucza tajnego `[a]` (to samo dotyczy parametru `(B)` i klucza `[b]`). Ma to na celu wzmocnienie niezawodności logarytmu dyskretnego (więcej informacji na ten temat znajdziesz w dalszej części rozdziału).

Inną właściwością potęgi modularnej jest możliwość zapisu `B^a` i `A^b (mod p)` w następujący sposób:

```
B ^a ≡ (g^b)^a (mod p)
A ^b ≡ (g^a)^b (mod p)
```

Tak więc w przypadku właściwości wykładnika modularnego otrzymujemy następujące równanie:

```
g^(b*a) ≡ g^(a*b) (mod p)
```

Spójrz na przykłady:

- Alicja: `(7^6)^3 ≡ 7^(6*3) ≡ 9 (mod 11)`
- Bartek: `(7^3)^6 ≡ 7^(3*6) ≡ 9 (mod 11)`

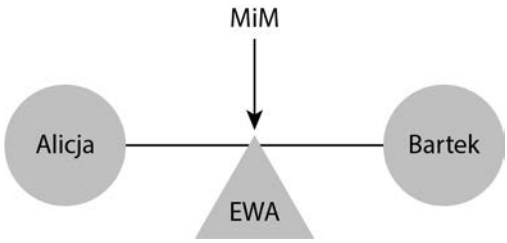
Jest to sztuczka matematyczna, dzięki której algorytm D-H może działać zgodnie z oczekiwaniami.

W następnym punkcie przedstawię słabe strony algorytmu D-H i ataki, które można na niego przeprowadzić.

Kryptoanaliza algorytmu D-H i potencjalnych ataków na niego

Najczęściej przeprowadzanym rodzajem ataku na algorytm D-H jest atak typu **man-in-the-middle** (**MiM**).

Z tego typu atakiem mamy do czynienia, gdy atakujący infiltruje kanał komunikacyjny oraz szpieguje, blokuje lub modyfikuje komunikację między nadawcą i odbiorcą. Atak zwykle jest przeprowadzany przez Ewę (atakujący) podszywającą się pod jedną z dwóch zaufanych stron komunikacji (rysunek 3.2).



Rysunek 3.2. Ewa przeprowadzająca atak typu man-in-the-middle

Przypomnij sobie, co się dzieje w *kroku 3.* algorytmu D-H: Alicja i Bartek wymienili się swoimi parametrami publicznymi, `(A)` i `(B)`.

Tutaj Alicja przesyła parametr `(A)` Bartkowi, który przesyła jej parametr `(B)`.

W tym momencie Ewa (atakujący) wtrąca się do komunikacji i podszywa pod Alicję.

Dlatego atak typu MiM przebiega w następujący sposób:

- **Krok 3.** Alicja przekazuje parametr `(A)` Bartkowi, który przekazuje jej parametr `(B)`.
- **W tym momencie przeprowadzany jest atak typu MiM.** Ewa przekazuje parametr `(E)` Bartkowi, który przekazuje jej parametr `(B)`, sądząc, że odbiorcą jest Alicja.

Przeanalizujmy funkcję Alicji, `A ≡ g^a (mod p)`, i Bartka, `B ≡ g^b (mod p)`. Ta operacja przekazania miałaby znaczenie krytyczne, gdyby została przeprowadzana w normalnej arytmetyce, a nie dla ciał skończonych.

Istnieje jeszcze inny możliwy rodzaj ataku, znany jako **atak urodzinowy**, który jest jednym z najsłynniejszych ataków przeprowadzanych na logarytmy dyskretne. Polega na ustaleniu, że w grupie osób co najmniej dwie będą obchodziły urodziny tego samego dnia, więc w grupie cyklicznej można znaleźć pewne takie same wartości (kolizje) pomocne w rozwiązaniu logarytmu dyskretnego.

Ważna uwaga

W tym miejscu `(E)` to parametr publiczny Ewy, który został wygenerowany przez jej klucz prywatny, a nie za pomocą szyfrowania.

Przechodząc do ostatniego fragmentu algorytmu, można zobaczyć efekt przeprowadzenia ataku typu MiM.

Załóżmy, że Alicja i Bartek używają algorytmu D-H do wygenerowania współdzielonego klucza prywatnego, który zostanie użyty do zaszyfrowania wiadomości.

Wiadomość od Alicji: *Bartek, przelej 10 000 zł na moje konto o numerze 1234567*.

Po wykonaniu *kroku 3.*, w którym Bartek i Ewa (podszywająca się pod Alicję) wymienili się parametrami publicznymi, Ewa wysyła Bartkowi zmodyfikowaną wiadomość (przechwyconą od Alicji), zaszyfrowaną za pomocą klucza współdzielonego.

Załóżmy, że zaszyfrowana wiadomość od Ewy ma następującą postać: `bu3fb3440r3nrunfjr3umi4gj57*je`.

Bartek otrzymuje tę zaszyfrowaną wiadomość (przekonany, że pochodzi ona od Alicji), deszyfruje ją za pomocą klucza współdzielonego D-H i otrzymuje w wyniku wiadomość w postaci zwykłego tekstu.

Po przeprowadzeniu ataku typu MiM Ewa zmodyfikowała wiadomość, która teraz brzmi: *Bartek, przelej 10 000 zł na moje konto o numerze 3217654*.

Jak można zauważyć, numer konta został zmieniony na należący do Ewy. Taki atak może mieć poważne skutki.

Analiza *kroku 3.* nie ma znaczenia krytycznego (jak już wspomniałem), ponieważ parametry `(A)` i `(B)` zostały przekazane w trybie nieszyfrowanym. Pytanie, które można sobie w tym miejscu zadawać, brzmi: skąd Bartek będzie miał pewność, że parametr `(A)` pochodzi od Alicji?

Odpowiedź na takie pytanie jest następująca: korzystając z algorytmu D-H, Bartek nie będzie miał pewności, że parametr `(A)` pochodzi od Alicji, a nie od Ewy (atakujący). Podobnie Alicja nie będzie miała pewności, że parametr `(B)` otrzymała od Bartka. W przypadku braku informacji dodatkowych dotyczących tożsamości obu stron komunikacji, algorytm D-H, bazując jedynie na otrzymanych parametrach, jest podatny na atak polegający na podszywaniu się pod innego użytkownika, czyli atak MiM.

Przedstawiony przykład wyraźnie pokazuje, że nadawca (Alicja) i odbiorca (Bartek) muszą mieć możliwość potwierdzenia swojej tożsamości. Konieczne jest również potwierdzenie, że ich klucze publiczne, `(A)` i `(B)`, pochodzą odpowiednio od Alicji i Bartka. Aby uniknąć problemu w postaci ataku typu MiM i mieć możliwość identyfikowania użytkowników kanału komunikacji, jedną z najczęściej stosowanych technik jest wykorzystanie podpisu cyfrowego. Dokładne omówienie podpisów cyfrowych znajdziesz w następnym rozdziale, który w całości poświęciłem tym metodom kryptograficznym.

Algorytm klucza prywatnego i publicznego może być używany do identyfikowania stron komunikacji i uniemożliwienia ataku typu MiM. W części IV tej książki przedstawię wybrane algorytmy klucza prywatnego i publicznego nowej generacji, które choć nie są asymetryczne, zapewniają wiele sposobów ich podpisywania.

Algorytm D-H można zaimplementować także za pomocą krzywych eliptycznych, co dokładniej przeanalizuję w rozdziale 7.

RSA

Wśród algorytmów kryptograficznych RSA można uznać za gwiazdę. Piękno tego algorytmu odpowiada jego logicznej prostocie i wewnętrznej sile, dzięki której mimo upływu 40 lat nadal jest stosowany do ochrony ponad 80% transakcji handlowych na świecie.

Nazwa tego algorytmu jest akronimem pochodzącym od nazwisk jego twórców — **Rivest**, **Shamir** i **Aldemar**. **RSA** to algorytm, który można określić mianem doskonałego algorytmu asymetrycznego. W 1997 roku angielska agencja kryptograficzna CESG przypisała Jamesowi Allisowi opracowanie w 1970 roku szyfrowania z użyciem klucza publicznego. Ta sama agencja uznała, że w 1973 roku Clifford Cocks napisał dokument przedstawiający podobną wersję algorytmu RSA.

Podstawowym założeniem algorytmu asymetrycznego jest używanie odmiennych kluczy szyfrowania i deszyfrowania.

Przypomnij sobie analogię z kłódkami, którą przedstawiłem, omawiając *algorytm Diffiego-Hellmana*. Wyjaśniłem wówczas, że każdy (a nie tylko Alicja i Bartek) może założyć kłódkę na metalowym pudełku. To jest prawdziwy problem związany z atakiem typu *man-in-the-middle*, ponieważ kłódka nie może być rozpoznana jako należąca do Bartka lub Alicji.

W celu rozwiązania tego problemu można przeprowadzić inny eksperyment myślowy, wykorzystując do tego podobną analogię, choć nieco inaczej określając sytuację.

Załóżmy, że Alicja wykonuje wiele kopii swojej kłódki i przekazuje je wszystkim urzędom pocztowym w kraju, zachowując w bezpiecznym miejscu klucz pozwalający otworzyć wszystkie te kłódki.

Jeżeli Bartek zechce wysłać Alicji tajną wiadomość, musi udać się na pocztę, poprosić o kłódkę Alicji, umieścić wiadomość w pudełku, a następnie je zamknąć na kłódkę.

Dzięki temu nadawca (Bartek) od chwili założenia kłódki nie będzie w stanie odszyfrować wiadomości. Natomiast Alicja po otrzymaniu pudełka będzie mogła je otworzyć za pomocą swojego bezpiecznie przechowywanego klucza.

W zasadzie w algorytmie RSA (w przeciwieństwie do algorytmu D-H) Bartek szyfruje wiadomość za pomocą klucza publicznego Alicji. Po zaszyfrowaniu wiadomości Bartek nie może jej zdeszyfrować, natomiast Alicja może ją zdeszyfrować za pomocą posiadanego klucza prywatnego.

To był pierwszy krok umożliwiający przeniesienie koncepcji szyfrowania asymetrycznego z czystej teorii do praktycznego zastosowania. Algorytm RSA pozwolił zaszyfrować wiadomość za pomocą klucza publicznego odbiorcy i odszyfrować ją za pomocą klucza prywatnego. Aby takie rozwiązanie było możliwe, RSA musi korzystać z konkretnej funkcji matematycznej, którą omówię w dalszej części rozdziału, podczas wyjaśniania sposobu działania algorytmu RSA.

Jak wcześniej wspomniałem, algorytm ten opracowało trzech naukowców. W owym czasie wszyscy trzej byli badaczami zatrudnionymi w MIT w Bostonie. Mówimy teraz o późnych latach siedemdziesiątych ubiegłego wieku. Po opracowaniu algorytmu D-H Ronald Rivest był niezwykle zafascynowany tym nowym rodzajem kryptografii. Najpierw zaangażował matematyka Leonarda Adlemana, a później innego kolegę z działu informatyki, Aida Shamira, który dołączył do zespołu. Celem Rivesta był opracowanie matematycznego sposobu przekazywania tajnych wiadomości zaszyfrowanych za pomocą klucza publicznego i deszyfrowanych z użyciem klucza prywatnego odbiorcy. Jednak w algorytmie D-H wiadomość mogła być szyfrowana jedynie po przekazaniu klucza, z użyciem tego samego klucza współdzielonego. Problem polegał na znalezieniu sposobu wysłania wiadomości zaszyfrowanej za pomocą klucza publicznego i deszyfrowanej z użyciem klucza prywatnego. Jak już wspomniałem, konieczne było zastosowanie konkretnej matematycznej funkcji odwrotnej. Jest to prawdziwa wartość dodana wynalezienia algorytmu RSA, którą wkrótce omówię.

Zgodnie z tym, co usłyszałem od Rivesta, historia tego odkrycia jest dość zabawna. W kwietniu 1977 roku Rivest i Adleman spotkali się w domu studenta przy okazji świąt Wielkanocy. Wypili trochę więcej wina i około północy Rivest wrócił do domu. Zaczął zastanawiać się nad problemem, który już od niemalże roku nie dawał mu spokoju. Leżąc w łóżku, otworzył książkę matematyczną i odkrył funkcję, która mogła okazać się doskonała do osiągnięcia celu postawionego grupie.

Odkryta przez niego funkcja była związana z problemem rozkładu na czynniki konkretną funkcją odwrotną w arytmetyce modularnej.

W rozdziale 1. wyjaśniłem już, że problem związany z rozkładem na czynniki ogromnej liczby poprzez mnożenie dwóch dużych liczb pierwszych jest uznawany za bardzo trudny do rozwiązania, nawet w przypadku komputerów dysponujących ogromną mocą obliczeniową.

Omówienie algorytmu RSA

W celu zrozumienia tego algorytmu rozważymy sytuację, w której Alicja i Bartek wymieniają się tajnymi wiadomościami.

Załóżmy, że Bartek chce wysłać Alicji tajną wiadomość. Oto przyjęte założenia:

- `M` — tajna wiadomość,
- `e` — parametr publiczny (zwykle stała liczba),
- `c` — szyfrogram lub kryptogram,
- `p,q` — dwie ogromne liczby pierwsze (klucze prywatne Alicji).

Zapoznaj się teraz z procedurami generowania klucza prywatnego i publicznego. Jak możesz zobaczyć, podstawą algorytmu RSA (i jego funkcją magiczną) jest wygenerowanie klucza prywatnego Alicji, [d].

Generowanie klucza:

Klucz publiczny Alicji, (N), jest dostępny za pomocą następującego fragmentu kodu:

```
N = p*q
```

Jak wcześniej wspomniałem, mnożenie dwóch ogromnych liczb pierwszych powoduje, że bardzo trudne jest przeprowadzenie rozkładu na czynniki wartości (N). Ogólnie rzecz biorąc, atakującemu będzie niezwykle trudno ustalić liczby [p] i [q]. Klucz prywatny Alicji, [d], powstaje za pomocą następującego kodu:

```
[d] * e ≡ 1 (mod[p-1]*[q-1])
```

Bartek przeprowadza szyfrowanie:

```
c ≡ M^e (mod N)
```

Bartek przekazuje Alicji szyfrogram, (c), który następnie może go ona zdeszyfrować za pomocą klucza prywatnego, [d]:

```
c^d ≡ M (mod N)
```

I na tym koniec!

> **Uwaga**
>
> Pogrubione elementy — **M**, **d**, **p** i **q** — w algor1ytmie są chronione i pozostają tajne.

Przykład liczbowy:

```
M = 88
e = 9007
p = 101
q = 67
N = 6767
```

Krok 1. Bartek przeprowadza szyfrowanie w następujący sposób:

```
88^9007 ≡ 6621 (mod 6767)
```

Alicja otrzymuje szyfrogram, tzn. `c = 6621`.

Krok 2. Alicja przeprowadza deszyfrowanie w następujący sposób:

```
9007* d ≡ 1 (mod (101-1)*(67-1))
d = 3943
6621^3943 ≡ 88 (mod 6767)
```

Jak widać, tajna wiadomość, [M] = 88, pochodzi z klucza prywatnego Alicji, [d] = 3943.

Analiza RSA

Jest wiele kwestii do wyjaśnienia, ale najważniejsze pozostaje zrozumienie, dlaczego funkcja używana do deszyfrowania (c) i otrzymania [M] działa.

```
M ≡ c^[d] (mod N)
```

To jest *krok 2.*, czyli mamy tutaj funkcję szyfrowania. Notacja została odwrócona przez umieszczenie [M] po lewej stronie.

Powód, dla którego takie rozwiązanie działa, kryje się w równaniu generowania klucza:

```
[d] * e ≡ 1 (mod [p-1]*[q-1])
```

Tutaj [d] to klucz prywatny Alicji. W teorii Eulera funkcja prawdopodobnie zostanie zweryfikowana, ponieważ liczby [p] i [q] są ogromne, a [M] to przypuszczalnie liczba względnie pierwsza (N). Jeżeli to równanie zostanie zweryfikowane, etap szyfrowania będzie można zmodyfikować i zapisać w następującej postaci:

```
(M^e)^d (mod N)
```

Wykorzystując właściwości potęg i teorię Eulera, otrzymujemy:

```
M^(e*d) (mod N)
de ≡ 1 (mod (p-1)*(q-1))
```

To samo można zapisać w postaci M^1 = M (mod N).

Dlatego poprzez wstawienie [d] na etapie deszyfrowania Alicja może otrzymać [M].

Konwencjonalne ataki na algorytm RSA

Wszystkie ataki omówione we wcześniejszej części podrozdziału są doskonale znane i dlatego można je określać mianem konwencjonalnych.

Trzy pierwsze metody ataku na algorytm RSA są powiązane z parametrem publicznym (mod N). W celu przeprowadzenia ataku na N = p*q atakujący może podjąć następujące kroki:

- wykorzystać *efektywny* algorytm rozkładu na czynniki pozwalający ustalić wartości p i q;
- użyć nowego algorytmu, który w określonych warunkach umożliwi znalezienie liczb;
- wykorzystać komputer kwantowy (w przyszłości) w celu szybszego przeprowadzenia rozkładu na czynniki.

Przeanalizujmy teraz te trzy możliwości:

- W pierwszym przypadku efektywny algorytm rozkładu na czynniki nie jest jeszcze znany. Do najczęściej stosowanych metod rozkładu zaliczamy:
  - algorytm ogólnego sita ciała liczbowego,
  - algorytm sita kwadratowego,
  - algorytm faktoryzacji Rho Pollarda.
- W drugim przypadku jeśli (n) jest liczbą cyfr N = p*q i atakujący zna pierwsze (n/4) cyfr lub ostatnie (n/4) cyfr [p] bądź [q], wówczas istnieje możliwość faktoryzacji (N) w efektywny sposób. Mimo to istnieje znikoma możliwość, że atakujący będzie znał te cyfry. Na przykład jeśli [p] i [q] mają 100 cyfr, a pierwsze (lub ostatnie) 50 cyfr [p] będzie znane, powstaje możliwość rozkładu na czynniki N.

  Więcej informacji na ten temat znajdziesz w omówieniu rozkładu na czynniki za pomocą metody Coppersmitha. Jak wyjaśnię w dalszej części rozdziału, z tą metodą jest związanych więcej przypadków, w których wykładnik (e) lub [d], a nawet zwykły tekst, [M], są zbyt krótkie.
- Jeżeli atakujący użyje komputera kwantowego, teoretycznie będzie istniała możliwość rozkładu na czynniki N w krótkim czasie dzięki użyciu algorytmu faktoryzacji Shora. Jestem przekonany, że w przyszłości pojawią się inne, znacznie efektywniejsze algorytmy kwantowe. Tę teorię dokładniej wyjaśnię w rozdziale 8., z którego dowiesz się nieco więcej na temat komputerów i obliczeń kwantowych oraz kryptografii Q.

Jeżeli mamy bardzo krótki fragment zwykłego tekstu, [M], i wykładnik, (e), również jest krótki, to algorytm RSA można będzie złamać. Wynika to stąd, że operacja potęgowania, M^e, pozostaje w ramach modulo N. Dlatego na tym etapie szyfrowania możemy powiedzieć, że mamy do czynienia z następującym równaniem:

```
M^e < N
```

W tym przypadku wystarczające będzie użycie e-tego pierwiastka kwadratowego (c) w celu znalezienia]M[.

Ważna uwaga

Użyłem tutaj odwrotnego nawiasu kwadratowego,]M[, aby wskazać na odszyfrowanie wiadomości.

Przykład liczbowy:

```
M= 2
e= 3
N = 77
2^3 ≡ 8 (mod 77)
```

Skoro `e = 3`, to poprzez przeprowadzenie prostej operacji obliczenia pierwiastka kwadratowego, √, można uzyskać wiadomość w postaci zwykłego tekstu:

```
8^(1/3) = 2
```

W tym przypadku mamy do czynienia z matematyką liniową, a nie z modularną.

Sposobem na pokonanie tego problemu jest wydłużenie wiadomości przez dodanie do niej losowo wybranych bitów. Taka metoda jest bardzo powszechna w kryptografii i cyberbezpieczeństwie i nosi nazwę **dopełnienia**.

Wprawdzie mamy różne sposoby dopełnienia, ale w tym miejscu skoncentruję się tylko na **dopełnieniu bitowym**. Jak już wyjaśniłem w rozdziale 1., kod ASCII można wykorzystać do konwersji tekstu na system dwójkowy, więc wiadomość, `[M]`, jest ciągiem tekstowym bitów. Jeżeli dodasz losowo wybrane bity (zwykle na końcu, choć można je umieścić także na początku), otrzymasz coś następującego:

```
... | 1011 1001 1101 0100 0010 011 1 0000 0000 |
```

Jak widać, bity przedstawione pogrubioną czcionką oznaczają dopełnienie.

Ta metoda może być używana w celu dopełniania wiadomości o dowolnej liczbie bitów długości, niekoniecznie musi to być liczba całkowita wyrażona w bajtach. Na przykład wiadomość 23-bitowa zostanie dopełniona 9 bitami, aby w ten sposób wypełnić blok 32-bitowy.

Teraz, po przedstawieniu dokładniejszych informacji na temat RSA i właściwości matematyki modularnej, mogę przejść do omówienia pierwszej interesującej aplikacji zaimplementowanej za pomocą tego algorytmu.

Zastosowanie algorytmu RSA do weryfikacji przestrzegania umów międzynarodowych

Załóżmy, że kraj Alfa chce monitorować dane sejsmograficzne kraju Beta, aby mieć pewność, że ten nie prowadzi na swoim terenie żadnych prób z bronią nuklearną. Pod powierzchnią kraju Beta został zainstalowany zestaw czujników mierzących aktywność sejsmiczną, której dane są rejestrowane i szyfrowane. Następnie tak wygenerowane dane są przekazywane do kraju Alfa za pomocą na przykład satelity.

Interesujący sposób zastosowania RSA przedstawia się następująco:

- Kraj Alfa, `(A)`, chce mieć pewność, że w kraju Beta, `(B)`, dane nie zostały zmodyfikowane.
- Kraj Beta chce sprawdzić dane przed ich wysłaniem (w celach szpiegowskich).

Dane zebrane z czujników nazwiemy [x], więc sposób działania protokołu można opisać w przedstawiony tutaj sposób.

Generowanie klucza:

1. Kraj Alfa wybiera parametry: (N = p*q), w postaci iloczynu dwóch ogromnych liczb pierwszych, i (e).
2. Kraj Alfa wysyła (N, e) do kraju Beta.
3. Kraj Alfa zachowuje w tajemnicy klucz prywatny, [d].

Protokół pozwalający sprawdzić, czy przeprowadzane są eksperymenty z bronią nuklearną, został opracowany w następujący sposób:

- **Krok 1.** Czujnik umieszczony głęboko w ziemi pobiera dane, [x], i przeprowadza ich szyfrowanie z użyciem klucza prywatnego, [d]:

  x^d ≡ y (mod N)

- **Krok 2.** Początkowo oba parametry, (x) i (y), zostały przekazane przez czujnik do kraju Beta, aby umożliwić potwierdzenie trafności informacji. Kraj Beta przeprowadza sprawdzenie, które można zapisać za pomocą następującego równania:

  y^e ≡ x (mod N)

- **Krok 3.** Po zakończonym sukcesem sprawdzeniu kraj Beta przekazuje (x, y) do kraju Alfa, który z kolei sprawdza wynik, (x):

  x ≡ y^e (mod N)

Ważna uwaga

(x) oznacza dane pochodzące z czujnika, podczas gdy [d] to klucz prywatny kraju Alfa przechowywany wewnątrz chronionego oprogramowania czujnika pobierającego dane.

Szyfrowanie odbywa się w przeciwny sposób do tradycyjnego działania RSA.

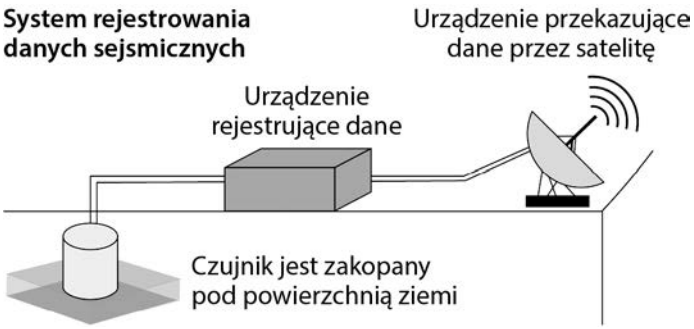
W przypadku y^e ≡ x (mod N) równanie będzie rozwiązane, a kraj Alfa może mieć pewność, że dane przekazane z kraju Beta są poprawne i nie doszło do modyfikacji wiadomości bądź majstrowania przy czujniku. Po prostu zaszyfrowana wiadomość, (x), odpowiadająca kryptogramowi, (y), może być wygenerowana jedynie przez osobę posiadającą klucz prywatny, [d].

Jeżeli w kraju Beta zostałaby podjęta próba manipulowania przy szyfrowaniu w pudełku przechowującym czujnik, to zmiana wartości (x) spowodowałaby, że kraj Beta miałby trudność z otrzymaniem zrozumiałej wiadomości w postaci zwykłego tekstu.

Jak wcześniej wspomniałem, w tym protokole mamy do czynienia z odwróconym RSA, a szyfrowanie odbywa się z zastosowaniem klucza prywatnego, [d], zamiast parametru publicznego, (e), jak to zwykle ma miejsce.

Trudność dla kraju Beta w modyfikacji tego szyfrowania w zasadzie sprowadza się do otrzymania zrozumiałej liczby bądź wiadomości. Próba modyfikacji kryptogramu, (y), nawet gdy parametr (x) był wcześniej znany, ma taką samą trudność obliczeniową jak w przypadku logarytmów dyskretnych (jak wcześniej wyjaśniłem, jest to bardzo trudny do rozwiązania problem).

W sposób graficzny proces ten pokazałem na rysunku 3.3.



Rysunek 3.3. Czujnik zakopany w ziemi, a pochodzące z niego dane są przekazywane za pomocą satelity

W ten sposób wyjaśniłem, jak algorytm RSA może posłużyć do sprawdzania realizacji umów międzynarodowych. W tym miejscu chciałbym przedstawić wprowadzenie do tematu, który będzie kontynuowany w rozdziale 6. i jest powiązany z niekonwencjonalnymi atakami na algorytm RSA i jego najbardziej znaną biblioteką, OpenSSL.

Ataki niekonwencjonalne

Omówione w tym punkcie ataki nazwałem *niekonwencjonalnymi*, ponieważ zostały zaimplementowane przeze mnie i aż do tej chwili nie zostały przetestowane ani opublikowane.

Jak pokażę w dalszej części książki, takie *ataki niekonwencjonalne* są możliwe nie tylko na algorytm RSA, ale również na inne algorytmy szyfrowania asymetrycznego. Przedstawione tutaj ataki zostały zaimplementowane w latach 2011 – 2014, mają na celu otrzymanie tajnej wiadomości, [M], bez znajomości klucza deszyfrowania Alicji, [d], a także tajnych liczb pierwszych, [p] i [q], stojących za (N). Wprawdzie te ataki przedstawię w tym punkcie, ale dokładne omówienie metod znajdziesz w rozdziale 6.

Część ataków przeprowadzanych na RSA może być zastosowana także względem większości omówionych w książce algorytmów szyfrowania asymetrycznego.

Nowy algorytm, który w przyszłości może pokonać problem rozkładu na czynniki, to *NextPrime*. Wywodzi się on z **algorytmu genetycznego** opracowanego przez mojego bliskiego przyjaciela, Gerarda Iovanego, który jego działanie wyjaśnił mi w 2009 roku. W swoim artykule *The Set of Prime Numbers* Iovane opisuje, jak można pobrać wszystkie liczby pierwsze za pomocą prostego algorytmu i korzystając ze wzorca odrzucać liczby niebędące pierwszymi.

Uwaga

Jeżeli chcesz dokładnie zapoznać się z algorytmem Gerarda Iovanego, przeczytaj artykuł *The Set of Prime Numbers*, który znajdziesz pod adresem *https://arxiv.org/abs/0709.1539*.

Po latach badań i wielu próbach opracowałem wreszcie funkcję matematyczną reprezentującą krzywą. Każde położenie na tej krzywej przedstawia liczbę pierwszą, a między wieloma położeniami znajdują się liczby półpierwsze (`N`) generowane przez dwie liczby pierwsze. Geometrycznie taka krzywa reprezentuje wszystkie liczby pierwsze we wszechświecie. Okazuje się, że położenie (`N`) zawsze znajduje się między miejscami na krzywej, w których znajdują się dwie liczby pierwsze, [`p`] i [`q`], a (`N`) jest w niemal równej odległości od ich położeń. Istnieje również możliwość pokazania, że liczby pierwsze mają wyraźną kolejność i nie są umieszczone losowo, bez zachowania kolejności, jak dotąd sądzono.

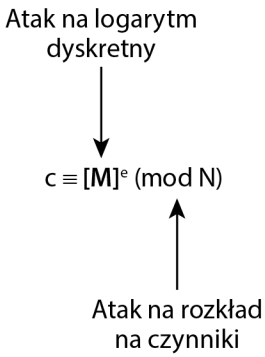
Odległość między dwoma obiektami liczb pierwszych, [`p`] i [`q`], mnożenia określającego wartość (`N`) jest odpowiednikiem liczby pierwszej znajdującej się między [`p`] i [`q`]. Na przykład *odległość liczb pierwszych* 17 i 19 wynosi zero, ta odległość między 1 i 100 wynosi 25, a między 10 000 i 10 500 jedynie 55.

Obecnie ten algorytm jest efektywny tylko w pewnych warunkach, np. gdy liczby [`p`] i [`q`] są *względnie blisko* siebie (w odległości wielomianowej). Jednak za interesujące można uznać, że nie ma tutaj znaczenia wielkość tych dwóch liczb pierwszych. Przeprowadziłem testy tego algorytmu z użyciem liczb pierwszych składających się z 10^{1000} cyfr.

Aby przypomnieć, jak ogromne to są liczby, warto w tym miejscu dodać, że 10^{80} reprezentuje liczbę cząsteczek we wszechświecie. Na przykład liczba półpierwsza składająca się z 10^{1000} cyfr odpowiada kluczowi publicznemu RSA o długości około 3000 bitów (obecnie to jeden z największych kluczy publicznych używanych w kryptografii). Jeżeli dwie liczby pierwsze są względnie blisko siebie, to użycie algorytmu NextPrime pozwala przetworzyć ten klucz w ciągu zaledwie kilku sekund.

W chwili powstawania książki pracowałem nad wersją algorytmu NextPrime bazującą na komputerze kwantowym. To może być następnej generacji algorytm rozkładu na czynniki za pomocą obliczeń kwantowych (podobnie do algorytmu faktoryzacji Shora, na którego temat więcej dowiesz się z rozdziału 8.).

Kontynuuję analizę pokazującą, jak można przeprowadzić atak na algorytm RSA. Jak widać na rysunku 3.4, w przypadku tego algorytmu mamy dwa punkty ataku: rozkład na czynniki (`N`) i potęgowanie dyskretne [`M^e`].



Rysunek 3.4. Punkty możliwego ataku na algorytm RSA

W przypadku większości analiz *konwencjonalnych* kryptolodzy koncentrują się na rozkładzie na czynniki (`N`), jak to wcześniej pokazałem. Jednak to nie jest jedyny potencjalny punkt ataku na ten algorytm. Drugim jest wykładnik, (`e`), a związany z tym problem dokładniej omówię w rozdziale 6. Te metody ataku to w zasadzie *tylne drzwi*, dzięki którym można odzyskać niezaszyfrowaną wiadomość, [`M`], bez znajomości tajnych parametrów nadawcy: [`d`], [`p`] i [`q`]. W rozdziale 6. podczas analizowania tych metod ataku wyjaśnię, że są one odpowiednikiem utworzenia tylnych drzwi w algorytmie RSA i jego bibliotece głównej, OpenSSL.

Zgodnie z omówionym tutaj paradygmatem RSA Bartek (nadawca) nie może zwrócić wiadomości po jej dostarczeniu. Jeżeli jednak będzie zastosowana niekonwencjonalna metoda ataku na RSA, ten paradygmat już nie obowiązuje. Bartek może utworzyć swoje „nieprawdziwe szyfrowanie" w celu zwrócenia wiadomości, [`M`], zaszyfrowanej za pomocą klucza publicznego Alicji, podczas gdy fałszywy szyfrogram może zostać zdeszyfrowany przez Alicję z wykorzystaniem etapu deszyfrowania RSA.

Czy ta metoda jest niepotrzebnym modelem nieprzedstawiającym rzeczywistych sytuacji, czy też znajduje zastosowania praktyczne?

Odpowiedź na to pytanie znajdziesz w rozdziale 6. Natomiast w kolejnym podrozdziale omówię jeszcze inny oparty na implementacji RSA protokół, który zyskał popularność: oprogramowanie PGP.

PGP

Pretty Good Privacy (**PGP**) to prawdopodobnie najczęściej używane oprogramowanie kryptograficzne na świecie.

Oprogramowanie PGP zostało opracowane przez Philipa Zimmermanna w trakcie zimnej wojny. Philip planował przenieść się z rodziną do Nowej Zelandii, wierząc, że w przypadku ataku nuklearnego ten odizolowany od reszty świata kraj będzie mniej narażony na skutki takiego ataku. Podczas planowania przeprowadzki do Nowej Zelandii coś spowodowało, że Philip zmienił zdanie i zdecydował się pozostać w USA.

Aby móc komunikować się z przyjaciółmi, Zimmermann, który był aktywistą antynuklearnym, opracował oprogramowanie PGP, umożliwiające bezpieczne przekazywanie wiadomości i plików przez internet. To oprogramowanie zostało wydane jako open source, bezpłatne do użycia niekomercyjnego.

W owym czasie ponad 40-bitowe systemy kryptograficzne były traktowane jako uzbrojenie. To oznacza, że kryptografia nadal jest uznawana za broń. Dlatego jeżeli zdecydujesz się na opatentowanie nowego systemu kryptograficznego, w USA wiąże się to ze spełnieniem wymogu, jakim jest uzyskanie pozwolenia z Departamentu Obrony, zanim nowy system będzie można opublikować. To stanowiło problem w przypadku oprogramowania PGP, w którym nie były stosowane klucze mniejsze niż 128-bitowe. Postępowanie karne i kara za złamanie tego wymogu są bardzo surowe i dlatego Zimmermann przez wiele lat miał *niewyjaśniony status prawny*, zanim rząd amerykański postanowił o zamknięciu dochodzenia przeciwko niemu i wycofaniu zarzutów.

PGP nie jest algorytmem, lecz protokołem. Innowacja polega tutaj na połączeniu szyfrowania symetrycznego i asymetrycznego. Protokół używa algorytmu szyfrowania asymetrycznego w celu przekazania klucza oraz szyfrowania symetrycznego do zaszyfrowania wiadomości i wygenerowania szyfrogramu. Ponadto wymagany jest podpis cyfrowy w celu identyfikowania użytkownika i uniknięcia ataku typu MiM.

Oto kroki protokołu PGP:

- **Krok 1.** Klucz zostaje przekazany za pomocą algorytmu szyfrowania asymetrycznego (ElGamal, RSA).
- **Krok 2.** Klucz przekazany z wykorzystaniem szyfrowania asymetrycznego staje się kluczem dla szyfrowania symetrycznego (DES, Triple DES, IDEA i AES, omówione w poprzednim rozdziale).
- **Krok 3.** Podpis cyfrowy jest używany do identyfikacji użytkowników (dokładne omówienie podpisu cyfrowego znajdziesz w następnym rozdziale).
- **Krok 4.** Deszyfrowanie odbywa się za pomocą klucza symetrycznego.

PGP to bardzo dobry protokół zapewniający wysoki poziom prywatności oraz zabezpieczający przekazywanie tajemnic handlowych.

Algorytm ElGamal

ElGamal to asymetryczna wersja algorytmu D-H. Ma ona pomóc w pokonaniu problemów związanych z atakiem typu MiM oraz brakiem możliwości zastosowania w algorytmie D-H podpisu cyfrowego potwierdzającego własność klucza. Ponadto ElGamal (podobnie jak RSA) to rzeczywisty algorytm szyfrowania asymetrycznego, ponieważ przeprowadza szyfrowanie wiadomości bez wcześniejszego przekazywania klucza.

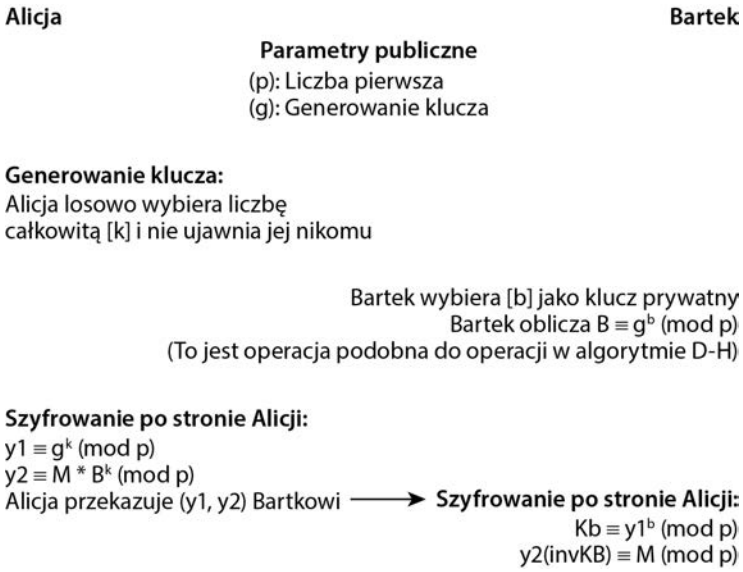
W omawianym algorytmie trudność wiąże się z rozwiązaniem problemu logarytmu dyskretnego. Jak się przekonasz w dalszej części rozdziału, istnieje również problem dotyczący rozkładu na czynniki.

ElGamal to pierwszy omówiony w tej książce algorytm, w którym pojawił się nowy element: losowo wybrana liczba całkowita, `[k]`, określona przez nadawcę i tajna. Ta liczba to ważny i innowacyjny element, dzięki któremu szyfrowanie staje się „efemeryczne", w tym sensie, że z powodu wartości `[k]` funkcja szyfrowania jest nieprzewidywalna. Ten nowy element można często spotkać w protokołach o zerowej wiedzy — do tego tematu jeszcze powrócę w rozdziale 5.

Zapoznaj się teraz z implementacją algorytmu ElGamal i zobacz, jak można go wykorzystać do przekazania tajnej wiadomości, `[M]`.

Alicja i Bartek to zawsze dwie osoby prowadzące komunikację. Alicja jest nadawcą, Bartek zaś jest odbiorcą wiadomości.

Na rysunku 3.5 pokazałem sposób działania algorytmu ElGamal.



Rysunek 3.5. Szyfrowanie i deszyfrowanie za pomocą algorytmu ElGamal

Jak można zobaczyć w ostatnim kroku deszyfrowania wiadomości przez Bartka, zastosowane zostało mnożenie odwrotne w (mod p). Ten rodzaj operacji to w zasadzie dzielenie przeprowadzane w ciele skończonym. Dlatego jeśli odwrotnością A jest B, to otrzymujemy A*B = 1 (mod p). Tutaj przedstawiłem przykłady pokazujące implementację tej odwróconej funkcji modularnej przeprowadzoną w programie Mathematica.

Po wyjaśnieniu sposobu działania algorytmu mogę przejść do zaprezentowania przykładu liczbowego, który powinien pomóc w zrozumieniu algorytmu.

Publicznie zdefiniowane parametry:

Publicznie zdefiniowanymi parametrami są p (ogromna liczba pierwsza) i g (generator):

```
p = 200003
g= 7
```

Generowanie klucza:

Alicja losowo wybiera dowolną liczbę całkowitą, [k], i jej nie ujawnia:

k = 23 (klucz prywatny Alicji)

Bartek generuje swój klucz publiczny, (B), rozpoczynając od klucza prywatnego, [b]:

k = 2357 (klucz prywatny Bartka)

```
B ≡ 7^2367 (mod 200003)
B = 151854
```

Szyfrowanie po stronie Alicji:

Alicja generuje tajną wiadomość, [M]:

```
M = 88
```

Następnie Alicja oblicza (y1, y2), dwa parametry publiczne, które zostaną przekazane Bartkowi:

```
y1 ≡ 7^23 (mod 200003)
y1 = 90914
y2 ≡ 88 * 151854^ 23 (mod 200003)
y2 = 161212
```

Alicja przekazuje te parametry Bartkowi: (y1 = 90914; y2 = 161212).

Szyfrowanie po stronie Bartka:

Bartek zaczyna od obliczenia (Kb) na podstawie wartości y1 = 90914 podniesionej do potęgi, którą jest jego klucz prywatny, [b] = 2367:

```
Kb ≡ 90914^2367 (mod 200003)
Kb = 10923
```

Odwrócona wartość `Kb` w `(mod p) [Reduce[Kb*x == 1, x, Modulus -> p]` (operacja przeprowadzona w programie Mathematica):

```
Inverted Kb ≡ 192331 (mod 200003)
```

Teraz Bartek może już zwrócić wiadomość, `[M]`.

Następnie Bartek mnoży `(y2)` przez `[Inverted Kb]`, a wartością zwrotną jest wiadomość, `[M]`:

```
y2* InvKb ≡ M (mod 200003)
```

Ostatecznym wynikiem operacji jest wiadomość `[M]`:

```
161212 * 192331 ≡ 88 (mod 200003).
```

Szyfrowanie ElGamal zostało wykorzystane w bezpłatnym oprogramowaniu **GNU Privacy Guard** (**GnuPG**). Na przestrzeni lat GnuPG zdobyło szeroką popularność i stało się faktycznym standardem wolnego oprogramowania przeznaczonego do zapewnienia prywatności komunikacji i obsługi podpisów cyfrowych. Podczas przekazywania kluczy kryptograficznych GnuPG używa najnowszej wersji PGP. Więcej informacji na temat tego oprogramowania znajdziesz na stronie *https://gnupg.org/software/index.html*.

Ważna uwaga

Jak wcześniej wspomniałem, przyjmuje się założenie, że za algorytmem ElGamal kryje się problem związany z logarytmem dyskretnym. Wynika to stąd, że jak już wyjaśniłem, parametry publiczne i klucze są definiowane przez równania oparte na logarytmach dyskretnych.

Na przykład `B ≡ g^b (mod p); Y1 ≡ g^k (mod p)` i `Kb ≡ y1^b (mod p)` to funkcje związane z logarytmem dyskretnym.

Wprawdzie problem dotyczący logarytmu dyskretnego jest uznawany za podstawowy problem zastosowany w algorytmie ElGamal, ale wykorzystano w nim także problem dotyczący rozkładu na czynniki. Powracamy teraz do funkcji szyfrowania, `(y2)`:

```
y2 ≡ M*B^k (mod p)
```

W tym przypadku mamy mnożenie. Dlatego atakujący może otrzymać wiadomość przez zastosowanie rozkładu na czynniki `(y2)`.

To stanie się wyraźnie widoczne po zredukowaniu funkcji:

```
H ≡ B^k (mod p)
```

Otrzymamy wówczas następującą postać:

```
y2 ≡ M* H (mod p)
```

Można ją również zapisać jako:

```
M ≡ y2/H (mod p)
```

Jak widać, (`y2`) to iloczyn [`M*H`]. Jeżeli komuś uda się znaleźć czynniki (`y2`), prawdopodobnie otrzyma również wiadomość [`M`].

Podsumowanie

W tym rozdziale przeanalizowałem podstawowe zagadnienia związane z szyfrowaniem asymetrycznym. W szczególności wyjaśniłem działanie logarytmu dyskretnego, a także wybranych spośród najbardziej znanych algorytmów zastosowanych w szyfrowaniu asymetrycznym, takich jak algorytm Diffiego-Hellmana, RSA i ElGamal. Omówiłem również interesujące zastosowania algorytmu RSA związane z przekazywaniem danych wrażliwych między dwiema nacjami. Z następnego rozdziału dowiesz się, jak można w sposób cyfrowy podpisywać te algorytmy.

Skoro omówiłem podstawy szyfrowania asymetrycznego, mogę przejść do wyjaśnienia zagadnienia podpisu cyfrowego. Jak już wspomniałem, omawiając PGP, te tematy są blisko ze sobą związane.

Algorytmy: poznaj serce kryptografii nowej generacji!

Naruszenie bezpieczeństwa systemu lub poufności danych to nie tylko kwestia utraty ważnych informacji, strat finansowych czy wizerunkowych. W skrajnych wypadkach może to być sprawa zdrowia i życia wielu ludzi. W świecie, w którym rozmaici przestępcy doskonalą swój arsenał, kryptografia i cyberbezpieczeństwo nabierają nowego znaczenia, a umiejętność efektywnej implementacji algorytmów kryptograficznych kolejnych generacji staje się cennym atutem.

Ta książka ułatwi studentom i inżynierom zrozumienie zasad działania algorytmów kryptograficznych następnej generacji. Przedstawiono w niej koncepcje algorytmów symetrycznych i asymetrycznych, jak również omówiono wszystkie nowoczesne techniki uwierzytelniania, przekazywania danych i wyszukiwania danych szyfrowanych. Wyjaśniono także techniki ochrony przed szpiegowaniem i hakerami. Zaprezentowano informacje o algorytmach Evolute o wiedzy zerowej, konsensusie w technologii blockchain, krzywych eliptycznych, kryptografii kwantowej i wyszukiwaniu homomorficznym. Nie zabrakło wyczerpującej prezentacji technik ataków i kryptoanalizy ważniejszych algorytmów stosowanych w informatyce.

W książce między innymi:

- kluczowe koncepcje kryptografii, algorytmy, protokoły i standardy
- efektywna implementacja algorytmów kryptograficznych
- nowe schematy i protokoły dla technologii blockchain i kryptowalut
- pionierskie algorytmy kryptografii kwantowej
- przeprowadzanie ataków na zaszyfrowane dane

Massimo Bertaccini jest badaczem i przedsiębiorcą. Zajmuje się kryptografią, cyberbezpieczeństwem i technologią blockchain. Razem z zespołem inżynierów opracował i zaimplementował pierwszy na świecie silnik wyszukiwania, który potrafi działać z danymi zaszyfrowanymi. Obecnie wykłada modele matematyczne na EMUNI University.

Helion

helion.pl

HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-289-0012-7

Cena: 79,00 zł

Packt