Bezpieczeństwo systemów informatycznych

Kryptografia

Marek Miśkiewicz

2026-01-04

UMCS Lublin

Definicje i cele kryptografii

Problem z jakim należy się zmierzyć

- Wiadomość jest przeznaczona wyłącznie dla Bolka — treść wiadomości nie może zostać ujawniona
- Wiadomość jest przesyłana publicznym kanałem
  - Alicja i Bolek spotkali się wcześniej
  - Alicja i Bolek nie spotkali się osobiście i nie mogą się spotkać
  - Alicja i Bolek wymieniają wiadomości w sposób ciągły
  - Alicja i Bolek wymieniają wiadomość jednorazowo
- Skąd Bolek ma wiedzieć, że nadawca jest właśnie Alicja
- Skąd Alicja ma wiedzieć, że przesyła wiadomość właśnie do Bolka
- Skąd Bolek ma wiedzieć, że to co otrzymał jest dokładnie tym co wysłała Alicja

Co to jest kryptografia?

- Etymologia: z greckiego "*kryptós*" (ukryty) i "*gráphein*" (pisać)
- Współczesna definicja: nauka o technikach zapewniających **bezpieczną komunikację** w obecności stron trzecich (potencjalnych przeciwników) w otwartym kanale komunikacyjnym
- Kryptografia jako dziedzina interdyscyplinarna:
  - matematyka - systemy kryptograficzne, mechanizmy stojące za systemami kryptograficznymi
  - informatyka - implementacja programowa
  - elektrotechnika - implementacja sprzętowa
  - fizyka - ograniczenia, kryptografia kwantowa

Główne cele kryptografii

Triada CIA

- **Poufność** (Confidentiality): ochrona informacji przed nieautoryzowanym dostępem
- **Integralność** (Integrity): zapewnienie, że informacje nie zostały zmodyfikowane w nieautoryzowany sposób
- **Dostępność** (Availability): zapewnienie, że uprawnieni użytkownicy mają dostęp do informacji, gdy jest to potrzebne

Ponadto:

- Uwierzytelnianie (Authentication): potwierdzenie tożsamości stron komunikacji
- Niezaprzeczalność (Non-repudiation): zapobieganie zaprzeczeniu wykonania konkretnej akcji
- Prywatność (Privacy): ochrona danych osobowych i zachowań użytkowników

Kryptografia a kryptoanaliza

- **Kryptografia**: projektowanie bezpiecznych systemów
- **Kryptoanaliza**: analiza i próby złamania systemów kryptograficznych

Wzajemna zależność sprawia, że postęp w kryptoanalizie prowadzi do udoskonalenia systemów kryptograficznych. Typowym przykładem jest kryptosystem DES (Data Encryption Standard)

DES → 3DES → AES

Podstawowe zasady nowoczesnej kryptografii

- **Zasada Kerckhoffsa**: bezpieczeństwo systemu powinno zależeć tylko od tajności klucza, nie od tajności algorytmu. Zawsze zakładamy, że komunikacja odbywa się otwartym kanałem i każdy ma dostęp do przesyłanych danych.
- **Zasada otwartego projektu** (Open Design Principle): systemy kryptograficzne powinny być publicznie znane i poddawane przeglądowi. (specyfikacja AES: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf)
- **Zasada najmniejszego uprzywilejowania**: ograniczenie dostępu do informacji tylko do niezbędnego minimum. Zasada ta stanowi, że każdy użytkownik, program lub proces powinien mieć dostęp tylko do tych zasobów i informacji, które są absolutnie niezbędne do wykonania jego zadań. Jest to związane z implementacją mechanizmów kontroli dostępu.

Kryptografia w kontekście współczesnych zagrożeń

Cyberprzestępczość i ataki cybernetyczne

- Ransomware: złośliwe oprogramowanie szyfrujące dane ofiary w celu wymuszenia okupu
  - Przykłady: WannaCry, NotPetya, Ryuk
  - Znaczenie silnego szyfrowania dla ochrony kopii zapasowych
- Phishing i ataki socjotechniczne: wykorzystanie manipulacji psychologicznej do zdobycia poufnych informacji
  - Spear phishing: ukierunkowane ataki na konkretne osoby lub organizacje
  - Rola kryptografii w uwierzytelnianiu i weryfikacji źródeł komunikacji

Cyberprzestępczość i ataki cybernetyczne

- Ataki DDoS (Distributed Denial of Service): przeciążenie systemów w celu uniemożliwienia dostępu
  - Wykorzystanie kryptografii do filtrowania ruchu sieciowego i uwierzytelniania żądań
- Ataki na łańcuch dostaw: kompromitacja oprogramowania lub sprzętu w procesie produkcji lub dystrybucji
  - Znaczenie podpisów cyfrowych i bezpiecznej dystrybucji oprogramowania

Masowa inwigilacja i ochrona prywatności

- Programy inwigilacji prowadzone przez agencje rządowe (np. ujawnione przez Edwarda Snowdena)
  - Znaczenie end-to-end encryption w ochronie prywatności komunikacji
- Gromadzenie danych przez korporacje technologiczne
  - Rozwój technik privacy-preserving computation (np. homomorficzne szyfrowanie)
- Systemy rozpoznawania twarzy i śledzenia lokalizacji
  - Technologie takie jak zero-knowledge proofs do weryfikacji bez ujawniania tożsamości

Bezpieczeństwo danych w erze chmury i IoT (Internet of Things)

- Wyzwania związane z przechowywaniem danych w chmurze
  - Szyfrowanie danych w spoczynku i podczas transmisji
  - Zarządzanie kluczami w środowiskach rozproszonych
- Bezpieczeństwo urządzeń IoT
  - Lekkie protokoły kryptograficzne dla urządzeń o ograniczonych zasobach
  - Zabezpieczanie komunikacji między urządzeniami IoT a chmurą

Zagrożenia związane z rozwojem technologii kwantowych

- Potencjalne złamanie wielu obecnie używanych systemów kryptograficznych
  - Zagrożenie dla RSA, ECC i innych algorytmów opartych na problemach faktoryzacji i logarytmu dyskretnego
- Rozwój kryptografii post-kwantowej
  - Nowe algorytmy odporne na ataki kwantowe (np. oparte na kratach, kodach, haszach)
- Kwantowa dystrybucja klucza (QKD) jako potencjalne rozwiązanie

Zagrożenia dla infrastruktury krytycznej

- Ataki na systemy energetyczne, wodociągowe, transportowe
  - Znaczenie bezpiecznych protokołów komunikacyjnych w systemach SCADA
- Bezpieczeństwo systemów finansowych i bankowych
  - Kryptografia w zabezpieczaniu transakcji i przechowywaniu danych finansowych

Nowe formy cyberzagrożeń

- Deepfakes i manipulacje medialne
  - Rola kryptograficznych technik weryfikacji autentyczności mediów
- Ataki na systemy autonomiczne (np. samochody autonomiczne)
  - Bezpieczna komunikacja i uwierzytelnianie w systemach czasu rzeczywistego
- Zagrożenia dla prywatności w erze big data i sztucznej inteligencji
  - Techniki privacy-preserving machine learning

Wyzwania regulacyjne i geopolityczne

- Różnice w podejściu do kryptografii między różnymi krajami
  - Debata nad “backdoorami” i dostępem dla organów ścigania
- Wpływ regulacji (np. GDPR, CCPA) na projektowanie systemów kryptograficznych
- Cyberbezpieczeństwo jako element konfliktów międzynarodowych
  - Ataki sponsorowane przez państwa i cyberszpiegostwo

Podstawowe pojęcia

Szyfrowanie i deszyfrowanie

Oznaczenia:

- M - tekst jawny (plaintext)
- C - tekst zaszyfrowany - kryptogram (ciphertext)
- K - klucz szyfrujący

Najogólniej mówiąc szyfrowanie i deszyfrowanie to proces przekształcania teksty jawnego w szyfrogram z wykorzystaniem klucza:

- szyfrowanie: $E(M, K) \longrightarrow C$
- deszyfrowanie: $D(C, K) \longrightarrow M$

Nie trzeba wspominać, że:

$$D(E(M, K), K) = M$$

Algorytm szyfrujący

Matematyczna procedura używana do szyfrowania i deszyfrowania. Nie jest to zazwyczaj pojedyncza operacja matematyczna, ale szereg różnych operacji na tekście jawnym lub szyfrogramie.

- **Szyfr Cezara**: $c = m + k \pmod{n}$ - k to klucz kryptograficzny
- **Szyfr afiniczny**: $c = am + b \pmod{n}$ - tu kluczem jest para liczb (a, b)
- **Data Encryption Standard** - mechanizm DES opiera się na wielokrotnym zastosowaniu permutacji i operacji XOR na danych z kluczem, które są podzielone na mniejsze bloki, a następnie przekształcane przy użyciu funkcji nieliniowych (S-boxów) w ramach 16 rund.
- **RSA**: $c = m^e \bmod n$, $m = c^d \bmod n$

Klucz kryptograficzny

- **Klucz kryptograficzny**: tajna informacja używana do kontrolowania procesu szyfrowania i deszyfrowania.
- **Długość klucza**: liczba bitów w kluczu, wpływająca na poziom bezpieczeństwa.
  - AES: 128 lub 256 bitów
  - RSA: 1024 - 4096 bitów

System kryptograficzny	Długość klucza (bit)	Bezpieczeństwo klasyczne (bit)
RSA	2048	112
ECDSA	256	128
AES	128	128
Crystal Kyber-512	6400	128
Crystal Kyber-768	9472	192
Crystal Kyber-1024	12544	256
FrodoKEM-640	76928	128

Klucz kryptograficzny

Przestrzeń kluczy - zbiór wszystkich kluczy dla danego kryptosystemu, przy ustalonej długości klucza. Dla klucza o długości n bitów:

$$\text{liczba kluczy: } k = 2^n$$

Należy być jednak ostrożnym przy określaniu rozmiaru przestrzeni kluczy:

- Szyfr Cezara: $k = 26$
- AES-256: $k = 2^{256} = 1.16 \cdot 10^{77}$
- RSA z kluczem 2048: $k = 2^{2048}$???

klucz kryptograficzny

Atak Brute-force - w przypadku ataków kryptograficznych polega na przeglądzie całej przestrzeni kluczy i decyzji o wyborze jednego z nich jako poprawnego. Jest to metoda niezwykle naiwna, ale pokazuje, że pewna klasa kryptosystemów nie gwarantuje 100% bezpieczeństwa. Należy mieć świadomość, że poza systemem Vernama, żaden kryptosystem nie gwarantuje absolutnego bezpieczeństwa.

W szerszym kontekście idea ataku Brute-Force podważa bezpieczeństwo większości systemów.

Atak Brute-Force

Załóżmy, że mamy klucz o długości 64 bitów. Dysponujemy maszyną zdolną do wykonywania 10^6 deszyfrowań na sekundę. Czas potrzebny na przejrzenie połowy wszystkich kluczy:

$$\frac{2^{64}}{10^6} \approx 9 \cdot 10^{12} \text{ sekund} \approx 300000 \text{ lat}$$

Ograniczenia:

- zasoby
- czas
- koszty

Kryptografia symetryczna

Kryptografia symetryczna

Kryptografia symetryczna to technika szyfrowania, w której do szyfrowania i deszyfrowania danych używa się **tego samego klucza**. Oznacza to, że zarówno nadawca, jak i odbiorca muszą posiadać dokładnie ten sam klucz, aby móc bezpiecznie wymieniać informacje.

1. **Nadawca** używa klucza, aby zaszyfrować wiadomość (czyli przekształcić ją w formę niezrozumiałą dla niepowołanych osób, tzw. szyfrogram).
2. **Odbiorca** używa tego samego klucza, aby odszyfrować wiadomość i odzyskać jej pierwotną treść.

Najważniejsze zalety kryptografii symetrycznej to jej **szybkość** i **efektywność** przy przetwarzaniu dużych ilości danych. Główna wada natomiast to konieczność **bezpiecznego przekazania klucza** między stronami – jeśli klucz zostanie przechwycony, cała komunikacja może być zagrożona.

Zanim pójdziemy dalej - XOR

```
A  B  A XOR B  (A XOR B) XOR A
-------------------------------
0  0    0          0
0  1    1          1
1  0    1          0
1  1    0          1   to jest dokładnie to co w kolumne B
```

Dlaczego operacja XOR jest taka ważna

$$(a \oplus b) \oplus b = a$$

- bity wiadomości: $m_1 m_2 m_3 \ldots$
- bity klucza: $k_1 k_2 k_3 \ldots$
- szyfrowanie $c_i = m_i \oplus k_i$
- deszyfrownie $m_i = c_i \oplus k_i$

Szyfr Vernama

Szyfr Vernama, znany także jako **szyfr jednorazowy (one-time pad)**, to algorytm szyfrowania, który wykorzystuje klucz o takiej samej długości jak tekst jawny. Klucz musi być losowy i używany tylko raz. Każdy znak tekstu jawnego jest przekształcany za pomocą operacji XOR z odpowiadającym mu znakiem klucza, co tworzy szyfrogram.

Główne cechy szyfru Vernama:

- **Bezpieczeństwo absolutne**: Jeśli klucz jest całkowicie losowy, tajny i nigdy nie używany ponownie, szyfr Vernama jest **niemożliwy do złamania**.
- **Jednorazowość klucza**: Klucz musi być używany tylko raz. Ponowne użycie klucza naraża system na ataki.
- **Losowość**: Klucz musi być naprawdę losowy, aby zapewnić pełną ochronę danych.

Szyfr Vernama

```
Tekst jawny: spotkanieo18
--------------------------------------------
Klucz: 43 13 d3 9c 49 b9 02 03 0e b8 82 96
Szyfr: 30 63 bc e8 22 d8 6c 6a 6b d7 b3 ae
--------------------------------------------
Klucz: 43 13 d3 9c 49 b9 02 03 0e b8 81 9c
Tekst jawny: spotkanieo22
```

Kryptografia asymetryczna

Kryptografia asymetryczna, znana również jako kryptografia z kluczem publicznym, opiera się na wykorzystaniu **dwóch matematycznie powiązanych kluczy**:

- klucza publicznego
- klucza prywatnego.

Przykłady: RSA, ECC

Kryptografia asymetryczna - cechy

1. Para kluczy:
   - Klucz publiczny: może być swobodnie udostępniany
   - Klucz prywatny: musi być utrzymywany w tajemnicy przez właściciela
2. Jednokierunkowość:
   - Wiadomość zaszyfrowana kluczem publicznym może być odszyfrowana tylko kluczem prywatnym
   - Wiadomość zaszyfrowana kluczem prywatnym może być zweryfikowana kluczem publicznym

Kryptografia asymetryczna - cechy

3. Podstawowe zastosowania:
   - Szyfrowanie: nadawca używa klucza publicznego odbiorcy do zaszyfrowania wiadomości
   - Podpis cyfrowy: nadawca używa swojego klucza prywatnego do podpisania wiadomości
4. Bezpieczeństwo:
   - Opiera się na trudności matematycznych problemów, takich jak faktoryzacja dużych liczb
   - Bezpieczeństwo zależy od utrzymania klucza prywatnego w tajemnicy

Kryptografia asymetryczna - cechy

5. Rozwiązanie problemu wymiany kluczy:
   - Eliminuje potrzebę bezpiecznego kanału do wymiany kluczy, co jest głównym problemem w kryptografii symetrycznej
6. Wydajność:
   - Wolniejsza niż kryptografia symetryczna, dlatego często używana w połączeniu z nią

Kryptografia sym. vs asym.

Kryptografia symetryczna

Jeden klucz do szyfrowania i deszyfrowania

Kryptografia asymetryczna

Klucz publiczny do szyfrowania, klucz prywatny do deszyfrowania

Klucz prywatny do podpisywania, klucz publiczny do weryfikacji

AES

AES (Advanced Encryption Standard) to popularny algorytm symetryczny, zatwierdzony przez amerykański instytut NIST w 2001 roku jako standard zastępujący starszy algorytm DES. AES jest powszechnie stosowany w różnych aplikacjach związanych z bezpieczeństwem danych, takich jak bankowość elektroniczna, komunikacja czy ochrona danych osobowych.

AES działa na blokach danych o rozmiarze **128 bitów** i używa kluczy kryptograficznych o długości **128, 192 lub 256 bitów**, co oznacza, że istnieją trzy różne wersje algorytmu. Proces szyfrowania w AES składa się z serii **rund**, których liczba zależy od długości klucza (10 rund dla klucza 128-bitowego, 12 rund dla 192-bitowego i 14 rund dla 256-bitowego).

Każda runda składa się z operacji takich jak:

1. **SubBytes** – zamiana bajtów według ustalonej tablicy (S-Box),
2. **ShiftRows** – przestawianie wierszy macierzy danych,
3. **MixColumns** – mieszanie kolumn w macierzy,
4. **AddRoundKey** – dodanie klucza rundy do przetwarzanych danych.

Dyfuzja i konfuzja

Dyfuzja i konfuzja to dwa fundamentalne koncepty w kryptografii, które zostały wprowadzone przez Claude'a Shannona w celu opisania, jak szyfrowanie powinno chronić dane przed analizą.

Dyfuzja

Dyfuzja odnosi się do procesu, w którym zmiana jednego bitu w danych wejściowych (np. w tekście jawnym) prowadzi do rozproszenia tej zmiany na wiele bitów w szyfrogramie. Innymi słowy, każdy bit tekstu jawnego powinien wpływać na wiele bitów szyfrogramu, a nawet mała zmiana w danych wejściowych powinna skutkować znaczącą zmianą w zaszyfrowanym tekście.

Celem dyfuzji jest utrudnienie analizy statystycznej, ponieważ statystyczne właściwości tekstu jawnego są ukryte w szyfrogramie. Dzięki temu nie można łatwo odgadnąć, które fragmenty danych wejściowych odpowiadają którym fragmentom szyfrogramu.

Przykład dyfuzji w AES: Operacje takie jak **MixColumns** i **ShiftRows** w AES pomagają w rozpraszaniu danych po macierzy, co wprowadza dyfuzję, czyli rozprzestrzenienie zmian na wiele bitów.

Atak z tekstem jawnym

Konfuzja

Konfuzja polega na skomplikowaniu relacji między kluczem szyfrującym a szyfrogramem. Celem jest to, aby zmiana klucza miała nieliniowy i nieprzewidywalny wpływ na szyfrogram. Dzięki temu nawet mała zmiana w kluczu powoduje dużą i złożoną zmianę w szyfrogramie, co utrudnia przeciwnikowi przeprowadzenie ataku na podstawie analizy zależności między kluczem a szyfrogramem.

Przykład konfuzji w AES: Operacja **SubBytes** w AES, gdzie każdy bajt jest zamieniany zgodnie z ustaloną nieliniową tablicą (S-Box), wprowadza konfuzję, ponieważ utrudnia ustalenie związku między kluczem a tekstem jawnym.

Tryby AES

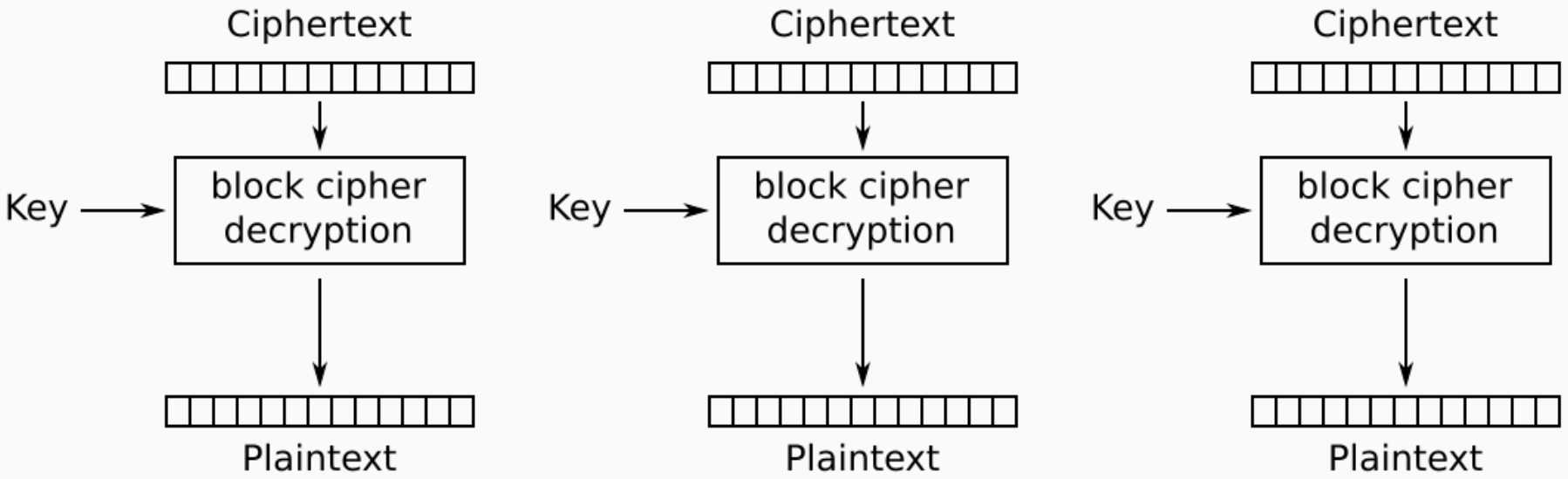
AES jest **szyfrem blokowym**, co oznacza, że dane wejściowe są dzielone na 128 bitowe bloki danych.

Tryby szyfrowania dla AES określają, w jaki sposób bloki danych są przetwarzane przez algorytm szyfrowania. Ponieważ AES działa na blokach o stałej długości (128 bitów), tryby szyfrowania umożliwiają bezpieczne przetwarzanie większych ilości danych, które mogą przekraczać ten rozmiar.

ECB

ECB (Electronic Codebook): Każdy blok danych jest szyfrowany niezależnie. Wadą jest brak ukrycia powtarzających się wzorców w danych, co może prowadzić do ich łatwego rozpoznania. Uważany za najmniej bezpieczny tryb.

ECB - deszyfrowanie

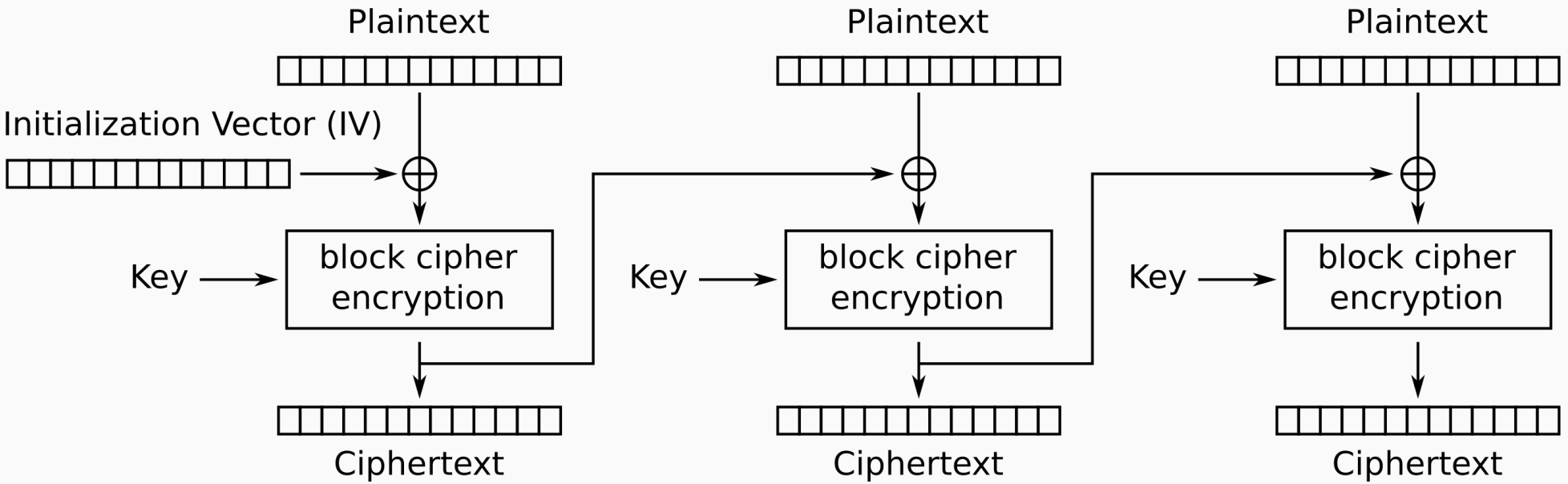


Electronic Codebook (ECB) mode decryption

CBC

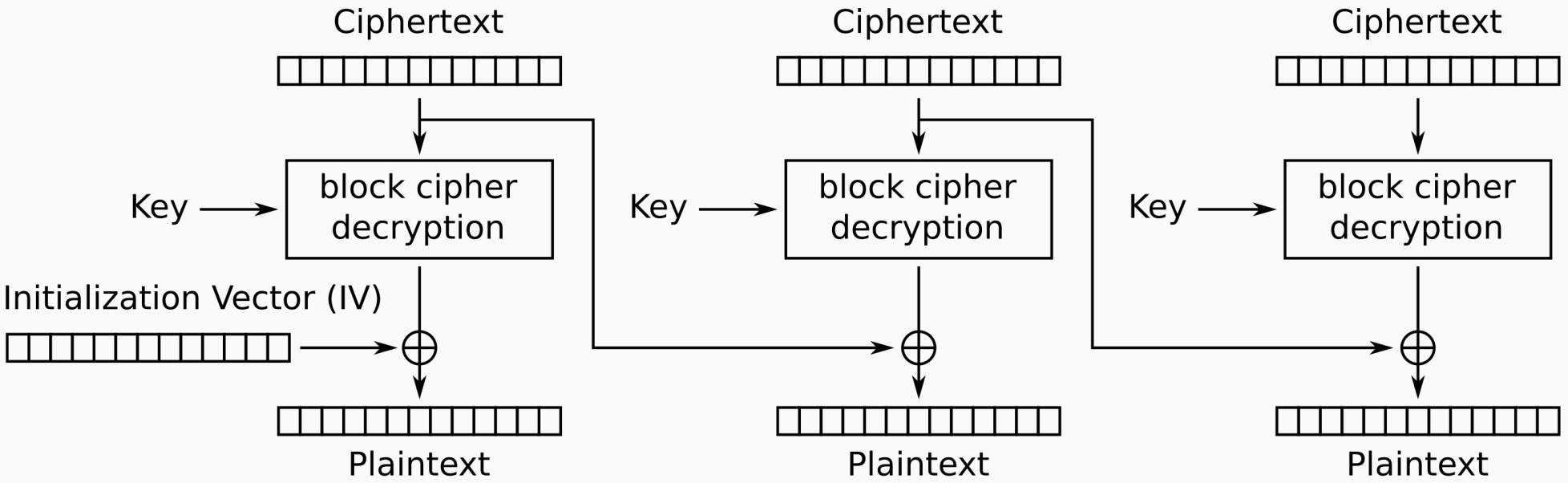
CBC (Cipher Block Chaining): Każdy blok danych jest połączony z poprzednim blokiem przed zaszyfrowaniem (dodanie operacją XOR). Pierwszy blok jest połączony z wektorem inicjującym (IV), co wprowadza losowość i zwiększa bezpieczeństwo.

CBC - szyfrowanie



Cipher Block Chaining (CBC) mode encryption

CBC - deszyfrowanie

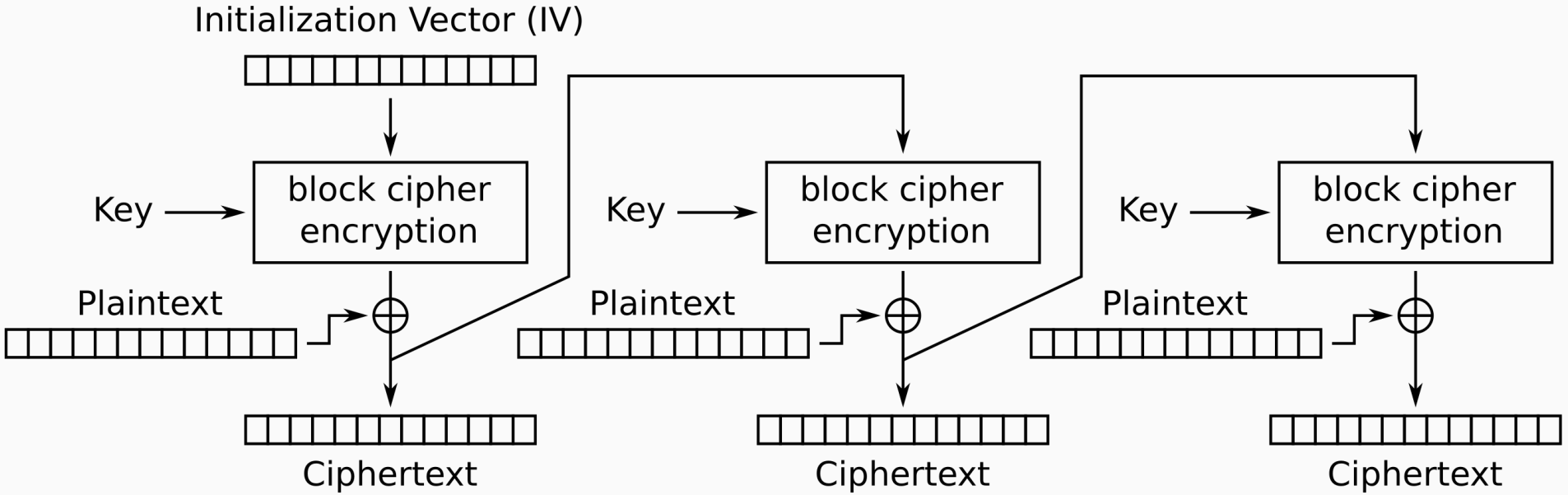


Cipher Block Chaining (CBC) mode decryption

CFB

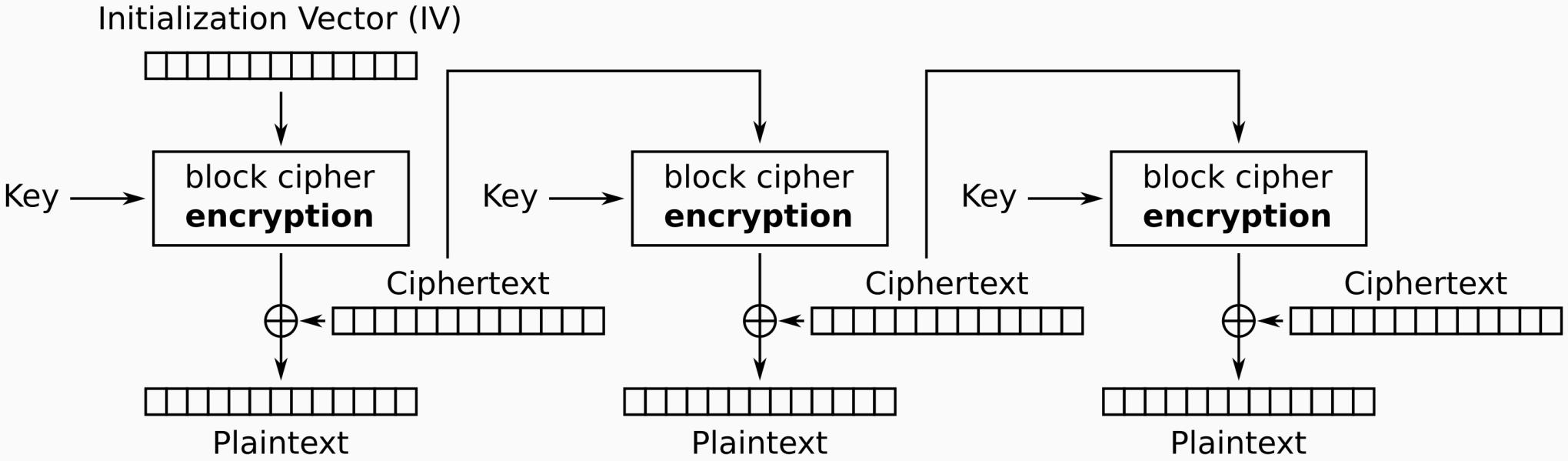
CFB (Cipher Feedback): Działa w sposób podobny do CBC, ale przetwarza dane jako strumień zamiast bloków, co umożliwia szyfrowanie ciągłe. Jest odporny na błędy, co oznacza, że drobne błędy w danych nie propagują się na cały szyfrogram.

CFB - szyfrowanie



Cipher Feedback (CFB) mode encryption

CFB - deszyfrowanie

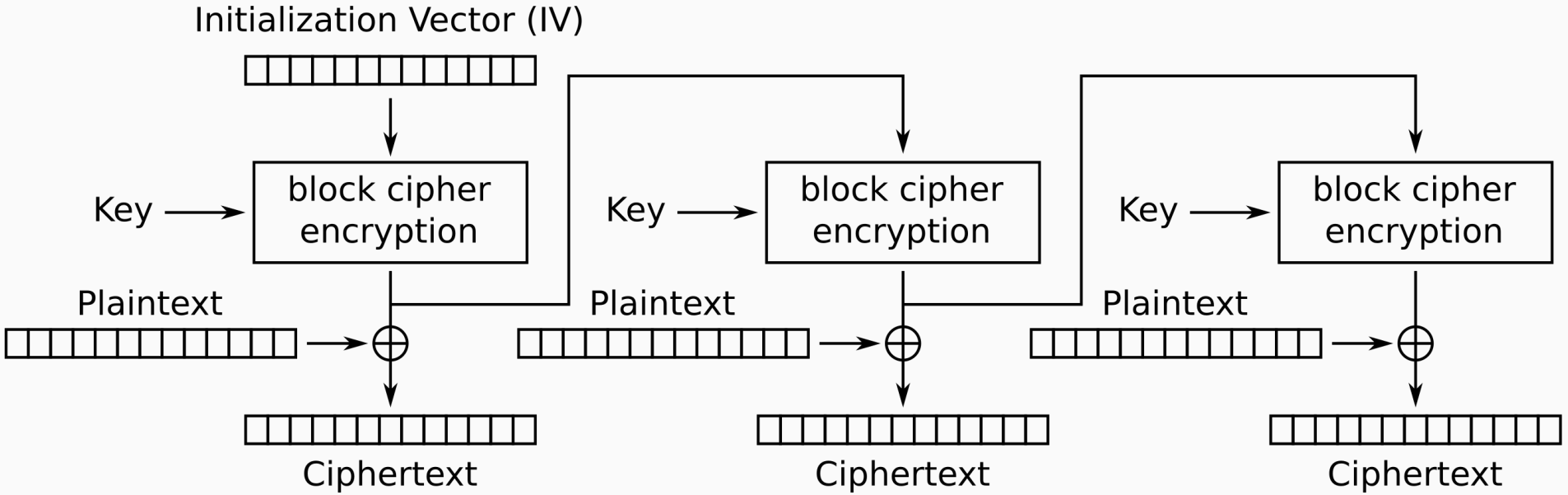


Cipher Feedback (CFB) mode decryption

OFB

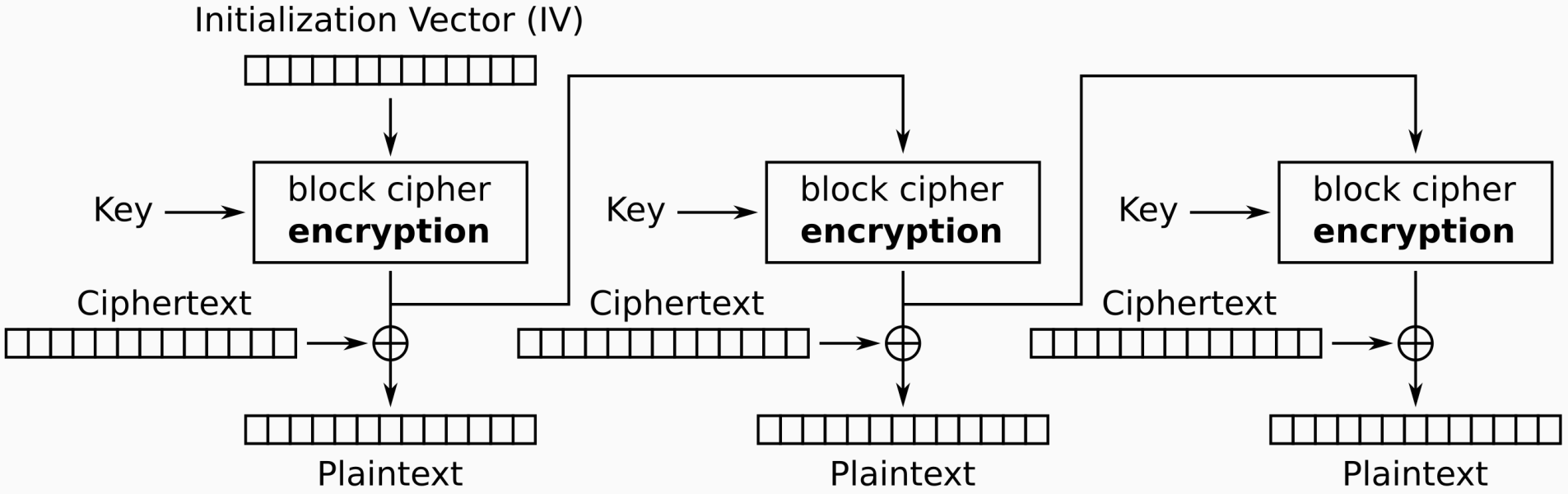
OFB (Output Feedback): Przetwarza dane jako strumień, podobnie jak CFB, ale generuje strumień klucza niezależnie od danych. Dzięki temu zmiana jednego bitu zaszyfrowanych danych nie wpływa na resztę szyfrogramu.

OFB - szyfrowanie



Output Feedback (OFB) mode encryption

OFB - deszyfrowanie

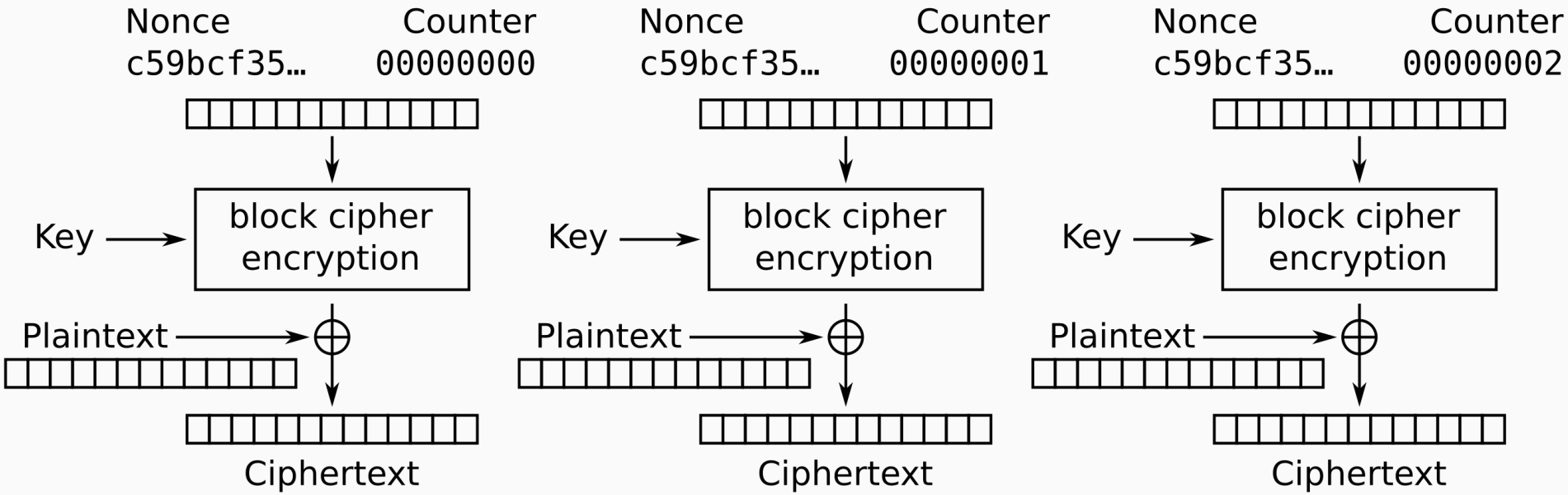


Output Feedback (OFB) mode decryption

CTR

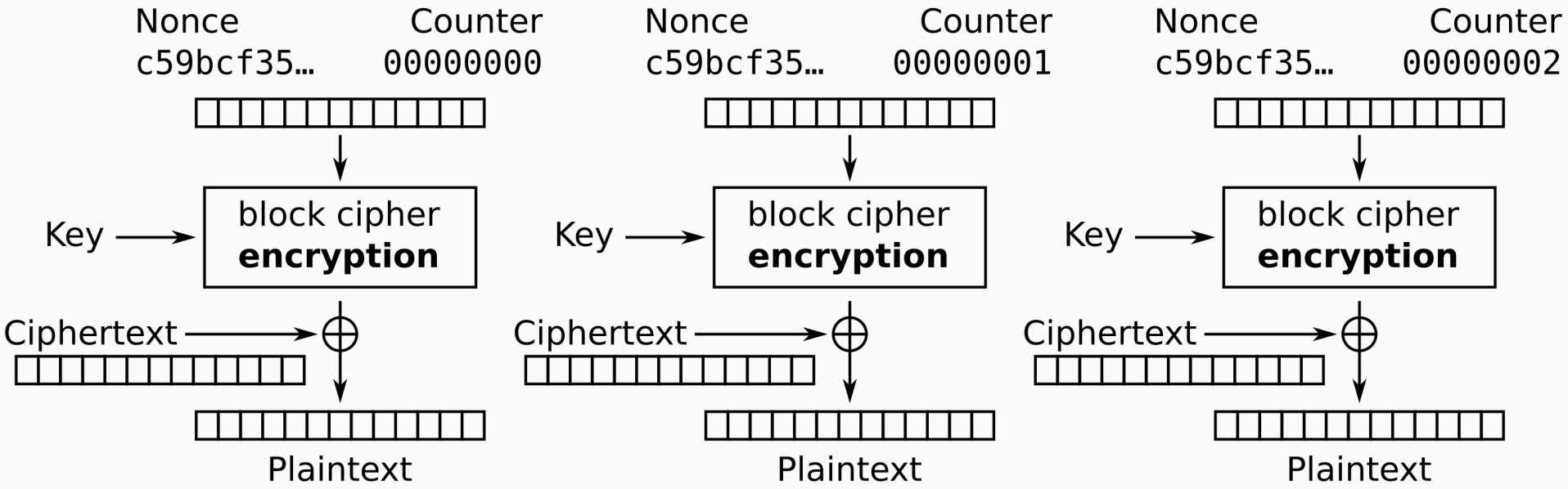
CTR (Counter): Każdy blok danych jest szyfrowany z wykorzystaniem licznika, który jest inkrementowany dla kolejnych bloków. CTR działa w trybie strumieniowym i jest bardzo efektywny, ponieważ operacje szyfrowania mogą być równolegle przetwarzane.

CTR - szyfrowanie



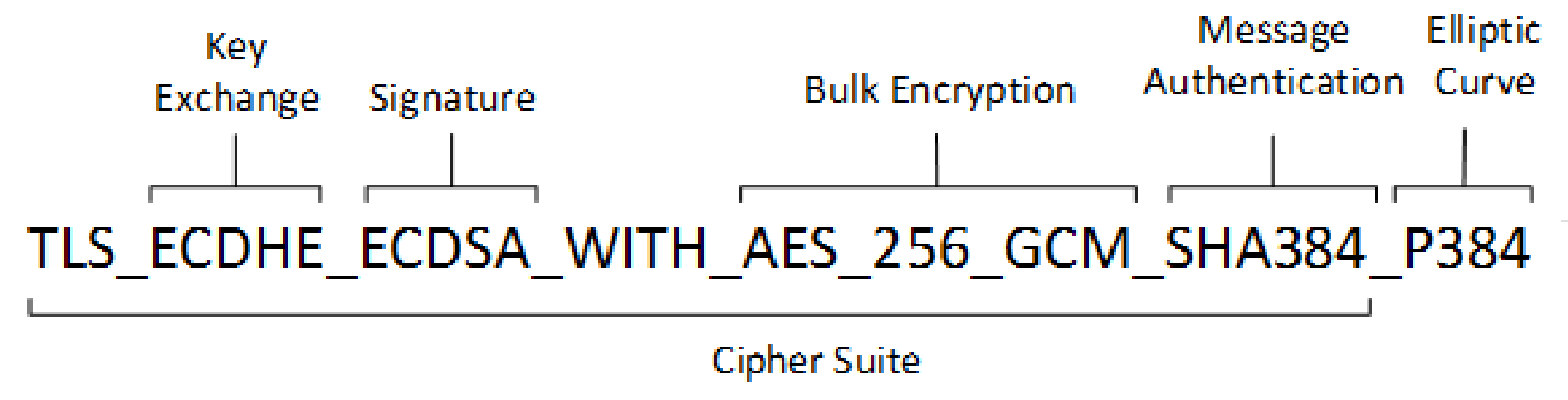
Counter (CTR) mode encryption

CTR - deszyfrowanie



Counter (CTR) mode decryption

Kryptografia symetryczna i asymetryczna



Funkcje skrótu

Funkcje skrótu

Definicja i właściwości

Funkcja skrótu to funkcja $H : \{0,1\}^* \to \{0,1\}^n$, która przekształca dane dowolnej długości w ciąg bitów o stałej długości n.

Właściwości kryptograficznych funkcji skrótu:

1. **Odporność na kolizje**: trudno znaleźć dwie różne wiadomości m_1 i m_2 takie, że $H(m_1) = H(m_2)$
2. **Odporność na pierwszą preimage**: mając skrót h, trudno znaleźć m takie, że $H(m) = h$
3. **Odporność na drugą preimage**: mając m_1, trudno znaleźć $m_2 \neq m_1$ takie, że $H(m_1) = H(m_2)$
4. **Efekt lawinowy**: niewielka zmiana w danych wejściowych powoduje radykalną zmianę skrótu

Funkcje skrótu

Popularne funkcje skrótu

- **MD5**: 128 bitów - przestarzały, podatny na kolizje
- **SHA-1**: 160 bitów - wycofywany, znaleziono praktyczne ataki
- **SHA-256/SHA-512**: rodzina SHA-2, aktualnie rekomendowana
- **SHA-3**: najnowszy standard NIST (Keccak)
- **BLAKE2/BLAKE3**: szybkie alternatywy

Zastosowania

- Weryfikacja integralności danych
- Przechowywanie haseł
- Podpisy cyfrowe
- Blockchain i proof-of-work
- HMAC (Hash-based Message Authentication Code)

Kryptografia asymetryczna

Kryptografia asymetryczna

Definicja

Kryptografia asymetryczna, znana również jako kryptografia z kluczem publicznym, wykorzystuje parę kluczy – publiczny i prywatny – dla zapewnienia bezpieczeństwa komunikacji i danych.

Podstawowy mechanizm: Użytkownik A może zaszyfrować wiadomość używając publicznego klucza Użytkownika B, który jedynie za pomocą swojego prywatnego klucza może ją odszyfrować. Użytkownik A może zweryfikować kluczem publicznym wiadomość podpisaną przez użytkownika B za pomocą jego klucza prywatnego.

Kluczowe różnice:

- **Klucze**: W asymetrycznej każdy ma swój klucz publiczny i prywatny, w symetrycznej obie strony używają tego samego klucza.
- **Zastosowania**: Kryptografia asymetryczna często używana do zabezpieczania sesji początkowych, symetryczna do szyfrowania większych strumieni danych.
- **Wydajność**: Kryptografia asymetryczna jest zazwyczaj wolniejsza ze względu na złożoność matematyczną, co sprawia, że jest mniej efektywna do szyfrowania dużych ilości danych.

Gdzie stosowana jest kryptografia asymetryczna

1. **SSL/TLS (Secure Sockets Layer / Transport Layer Security)** - protokół bezpieczeństwa stosowany do zabezpieczania komunikacji w Internecie, na przykład przy przeglądaniu stron WWW, wysyłaniu e-maili czy korzystaniu z usług bankowych online.
2. **SSH (Secure Shell)** - protokół umożliwiający bezpieczne zdalne logowanie i inne bezpieczne usługi sieciowe.
3. **PGP/GPG (Pretty Good Privacy / GNU Privacy Guard)** - służy do zabezpieczania komunikacji e-mailowej poprzez szyfrowanie i tworzenie elektronicznych podpisów.
4. **IPsec (Internet Protocol Security)** - protokół zapewniający bezpieczeństwo na poziomie warstwy IP, stosowany między innymi do VPN (Virtual Private Networks).

Gdzie stosowana jest kryptografia asymetryczna

5. **Blockchain i Kryptowaluty** - technologia blockchain oraz kryptowaluty, takie jak Bitcoin czy Ethereum. Klucze publiczne i prywatne służą do tworzenia portfeli oraz podpisywania transakcji, co zapewnia ich bezpieczeństwo i niezaprzeczalność.
6. **WebAuthn (Web Authentication)** - standard uwierzytelniania dla stron internetowych, wspierany przez wiele nowoczesnych przeglądarek.
7. **EIDAS (Electronic Identification, Authentication and Trust Services)** - europejski system identyfikacji cyfrowej i usług zaufania dla transakcji elektronicznych.
8. **X.509 (Standard Certyfikatów)** - standard używany do definiowania formatu certyfikatów cyfrowych. Jest stosowany w wielu systemach i protokołach bezpieczeństwa, takich jak SSL/TLS, podpis elektroniczny, S/MIME (Secure/Multipurpose Internet Mail Extensions) i inne.

RSA - Rivest Shamir Adleman

- **RSA** to jeden z pierwszych i najbardziej rozpowszechnionych algorytmów kryptografii asymetrycznej.
- Generowanie kluczy: Podstawą do generowania kluczy jest wybór dwóch dużych liczb pierwszych, obliczenie iloczynu

$$n = p \times q$$

oraz funkcji Eulera

$$\varphi(n) = \varphi(pq) = \varphi(p)\varphi(q) = (p-1)(q-1)$$

n jest elementem klucza publicznego wraz z eksponentem e. Znajomość rozkładu n na czynniki pozwala w łatwy sposób odtworzyć klucz prywatny.

$$ed \equiv 1 \bmod \varphi(n)$$

Klucze

- **Klucz publiczny**: para liczb n i e
- **Klucz prywatny**: para liczb $\varphi(n)$ i d

Szyfrowanie i deszyfrowanie

- szyfrowanie: $c = m^e \bmod n$
- deszyfrowanie: $m = c^d \bmod n$ (lub p i q)

Zauważmy, że:

$$m = c^d \bmod n = (m^e)^d \bmod n = m^{ed} \bmod n$$

RSA klucze openssl

```
Sams-MacBook-Pro-3:~ sambowne$ openssl rsa -in key2.pem -out public2.pem -pubout
writing RSA key
Sams-MacBook-Pro-3:~ sambowne$ openssl rsa -pubin -text < public2.pem
Modulus (2048 bit):
    00:de:35:30:7f:3f:1c:8a:b1:24:68:0e:7c:57:bc:
    31:f9:64:b4:f3:b1:e3:52:cf:df:d4:d6:b3:fd:bb:
    4b:c5:0f:6f:83:0f:da:53:af:1f:e1:33:bd:c4:aa:
    b6:fd:72:6c:ac:dd:6e:40:29:fe:65:7f:90:3f:38:
    c8:92:b4:39:21:00:bf:27:f0:af:2f:51:a2:e7:3f:
    2e:ba:01:b5:be:1e:48:78:a4:25:4c:5e:b4:8f:32:
    63:57:49:2e:7f:26:85:07:2c:af:f2:7a:14:dd:76:
    26:f5:a1:66:0c:f7:57:af:36:cc:5e:12:fa:d2:2f:
    b2:3e:87:a2:5d:a1:80:d4:c4:d9:de:a9:d4:50:8a:
    3f:69:12:e1:39:ca:8b:1e:68:7b:19:e2:19:18:84:
    1a:c9:f4:9f:53:e3:c7:62:68:79:2e:3f:f8:89:59:
    96:ee:d7:e2:38:29:36:a4:22:fe:b3:9e:92:40:4a:
    ec:74:62:38:4d:77:ba:cb:44:01:d8:aa:c8:76:33:
    e9:50:de:9a:43:12:ab:a0:08:ca:b1:2d:16:4f:9d:
    b4:02:4c:3a:88:2f:7e:69:bd:c7:27:81:b4:98:7a:
    8e:81:de:1b:30:c6:69:a5:20:0a:91:71:40:03:9c:
    56:2e:cb:4a:79:1b:b0:bd:53:6f:d1:d1:db:3a:38:
    ba:e1
Exponent: 65537 (0x10001)
writing RSA key
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA3jUwfz8cirEkaA58V7wx
+WS087HjUs/f1Naz/btLxQ9vgw/aU68f4TO9xKq2/XJsrN1uQCn+ZX+QPzjIkrQ5
IQC/J/CvL1Gi5z8uugG1vh5IeKQlTF60jzJjV0kufyaFByyv8noU3XYm9aFmDPdX
rzbMXhL60i+yPoeiXaGA1MTZ3qnUUIo/aRLhOcqLHmh7GeIZGIQayfSfU+PHYmh5
Lj/4iVmW7tfiOCk2pCL+s56SQErsdGI4TXe6y0QB2KrIdjPpUN6aQxKroAjKsS0W
T520Akw6iC9+ab3HJ4G0mHqOgd4bMMZppSAKkXFAA5xWLstKeRuwvVNv0dHbOji6
4QIDAQAB
-----END PUBLIC KEY-----
Sams-MacBook-Pro-3:~ sambowne$
```

RSA - atak

Bezpieczeństwo RSA bazuje na złożoności obliczeniowej problemu faktoryzacji dla dużych liczb.

Najszybszy znany algorytm faktoryzacji liczb całkowitych to algorytm faktoryzacji liczbowego pola sita (Number Field Sieve, NFS), który jest stosowany do rozkładania bardzo dużych liczb na czynniki pierwsze, zwłaszcza używanych w kryptografii, takich jak klucze RSA.

Złożoność czasowa algorytmu NFS: Złożoność algorytmu Number Field Sieve dla ogólnego przypadku można wyrazić jako:

$$L_n\left(\frac{1}{3}, c\right) = e^{(c+o(1))(\log n)^{\frac{1}{3}}(\log\log n)^{\frac{2}{3}}}$$

c jest stałą zależną od szczegółów implementacji algorytmu, a $o(1)$ oznacza składnik malejący do zera, gdy n rośnie do nieskończoności.

RSA - algorytm Shora

Algorytm Shora

Algorytm Shora, opracowany przez Petera Shora w 1994 roku, to kwantowy algorytm służący do faktoryzacji dużych liczb całkowitych, który może potencjalnie złamać systemy kryptograficzne oparte na kryptografii asymetrycznej, takie jak RSA. Jego znaczenie wynika z tego, że może on teoretycznie działać znacznie szybciej niż najlepsze znane algorytmy klasyczne do faktoryzacji, takie jak Number Field Sieve.

Złożoność czasowa algorytmu Shora

Algorytm Shora działa w czasie wielomianowym względem liczby cyfr liczby do faktoryzowania, co jest dużą poprawą w porównaniu z eksponencjalnym czasem działania klasycznych algorytmów. Dokładna złożoność to

$$L(n) = O\Big((\log n)^2(\log\log n)(\log\log\log n)\Big),$$

gdzie n to faktoryzowana liczba.

Obecny stan: https://www.nature.com/articles/s41598-021-95973-w

RSA OAEP

Zakłada się, że klucze o długości poniżej **1024** bitów nie gwarantuja dostatecznego bezpieczeństwa dla RSA.

OAEP

Optimal Asymmetric Encryption Padding (OAEP) jest zaawansowanym schematem paddingu używanym w algorytmach kryptografii asymetrycznej, takich jak RSA. Został opracowany, aby zwiększyć bezpieczeństwo szyfrowania przez dodanie dodatkowej warstwy zabezpieczeń, które chronią przed różnego rodzaju atakami, w tym przed atakami z wybranym tekstem jawnym.

W dużym uproszczeniu OAEP łączy wiadomość z losowym ciągiem bitów, dzięki czemu nawet wielokrotne szyfrowanie tej samej wiadomości daje różne szyfrogramy.

Uproszczony schemat OAEP

Założenia

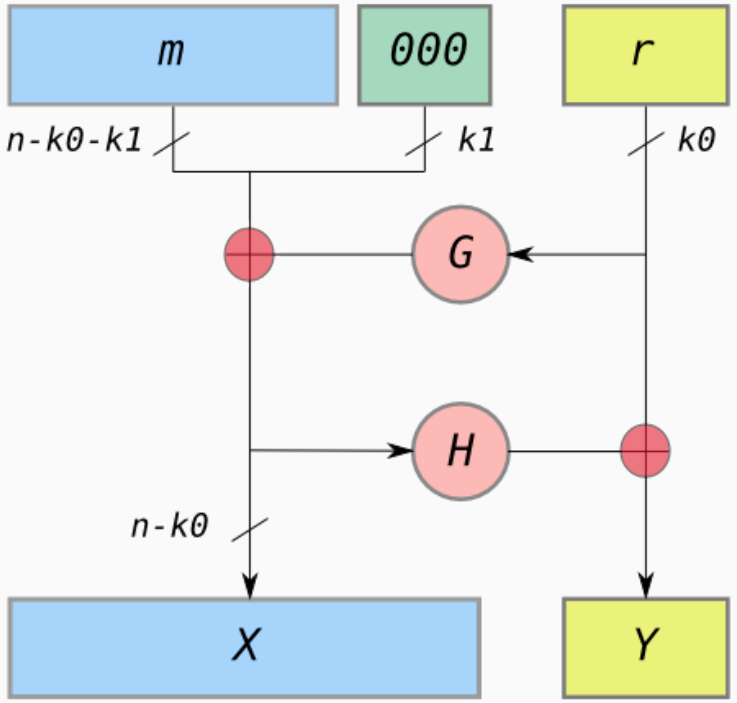
- m - wiadomość
- G i H - funkcje haszujące

Algorytm szyfrowania

1. Łączymy wiadomość m z ciągiem k_1 zer
2. Generujemy losowy ciąg bitów r
3. Wykonujemy $G(r)$
4. $X = m000...0 \oplus G(r)$
5. $Y = H(X) \oplus r$

Algorytm deszyfrowania

1. $r = Y \oplus (H(X))$
2. $m000...0 = X \oplus G(r)$



https://pl.wikipedia.org/wiki/Plik:Oaep-diagram-20080305.png

RSA - podpis

Założenia:

- klucz publiczny: (n, e)
- klucz prywatny: (p, q, d)
- wiadomość: m
- funkcja skrótu $H()$, $H(m) < n - 1$

RSA - podpis

Podpis

1. Generowanie skrótu wiadomości: $H(m)$
2. Generowanie podpisu $d_m = (H(m))^d \bmod n$
3. Podpisana wiadomość to para m i d_m

Weryfikacja

1. Generowanie skrótu z wiadomości: $H'(m)$
2. Weryfikacja podpisu: $\left(d_m\right)^e \bmod n = \left(H(m)^d\right)^e \bmod n = (H(m))^{ed} \bmod n = H(m)$
3. Porównanie czy $H'(m) = H(m)$

Czy tutaj nie ma miejsca szyfrowanie kluczem prywatnym a odszyfrowanie kluczem publicznym?

RSA - podpis

Kluczowe aspekty:

- **Autentyczność:** Weryfikacja podpisu potwierdza, że wiadomość została podpisana przez osobę posiadającą odpowiedni klucz prywatny, co stanowi potwierdzenie tożsamości nadawcy.
- **Integralność:** Zgodność skrótu wiadomości potwierdza, że treść wiadomości nie została zmieniona od czasu jej podpisania.
- **Nezaprzeczalność:** Nadawca nie może zaprzeczyć autorstwu wiadomości, gdyż podpis został stworzony jego unikalnym kluczem prywatnym.

Schemat Diffie-Hellmana

Schemat Diffie-Hellmana to jedna z pierwszych metod umożliwiających bezpieczną wymianę kluczy kryptograficznych przez niezabezpieczony kanał komunikacyjny. Został opracowany przez Whitfielda Diffiego i Martina Hellmana w 1976 roku. Schemat ten pozwala dwóm stronom, które wcześniej nie wymieniały się żadnymi tajnymi danymi, ustalić wspólny, tajny klucz, który może być następnie używany do zaszyfrowanej komunikacji.

Jak działa schemat Diffie-Hellmana

1. **Wybór parametrów publicznych:**
   - p - duża liczba pierwsza,
   - g - generator (tzw. pierwiastek pierwotny modulo p), który jest liczbą, dla której każda liczba mniejsza od p i względnie pierwsza z p jest resztą z potęgowania g modulo p.

Te parametry są publiczne i mogą być znane przez wszystkich uczestników komunikacji.

2. **Generowanie kluczy prywatnych i publicznych:**
   - Alice i Bob niezależnie wybierają swoje losowe, tajne klucze prywatne a oraz b (liczby mniejsze od p).
     - Następnie obliczają swoje publiczne klucze:
       - Alice oblicza $A = g^a \bmod p$ i wysyła A do Boba,
       - Bob oblicza $B = g^b \bmod p$ i wysyła B do Alice.

Jak działa schemat Diffie-Hellmana

3. **Obliczanie wspólnego klucza:**
   - Po otrzymaniu publicznego klucza od drugiej strony, zarówno Alice jak i Bob mogą obliczyć wspólny sekret, który posłuży jako klucz do zaszyfrowanej komunikacji:
     - Alice otrzymuje B od Boba i oblicza wspólny klucz s jako $s = B^a \bmod p$,
     - Bob otrzymuje A od Alice i oblicza wspólny klucz s jako $s = A^b \bmod p$.

Dzięki własnościom potęgowania modulo p, obliczenia Alice i Boba prowadzą do tego samego wyniku: $g^{ab} \bmod p$, co stanowi ich wspólny klucz.

Bezpieczeństwo schematu Diffie-Hellmana

Schemat Diffie-Hellmana opiera się na trudności obliczeniowej problemu logarytmu dyskretnego, który polega na znalezieniu x w równaniu $g^x \bmod p = A$, gdy znane są g, p, i A. Obecnie nie istnieje efektywny algorytm, który mógłby rozwiązać ten problem w rozsądnym czasie dla odpowiednio dużych i odpowiednio dobranych liczb p i g.

Jednakże, schemat Diffie-Hellmana jest podatny na ataki typu "man-in-the-middle", gdzie atakujący może przechwycić i zastąpić klucze publiczne, wysyłając swoje własne.

W praktycznych implementacjach stosowane są dodatkowe mechanizmy uwierzytelniania i zabezpieczenia, takie jak certyfikaty cyfrowe.

Krzywe eliptyczne

Krzywe eliptyczne

Kryptografia krzywych eliptycznych (Elliptic Curve Cryptography, ECC) to forma kryptografii asymetrycznej oparta na matematyce krzywych eliptycznych nad skończonymi ciałami. Wprowadzona w praktykę przez Victora S. Millera i Neala Koblitza w połowie lat 80. XX wieku, szybko zyskała popularność dzięki swojej wydajności i bezpieczeństwu przy stosunkowo mniejszych rozmiarach kluczy w porównaniu do tradycyjnych metod, takich jak RSA.

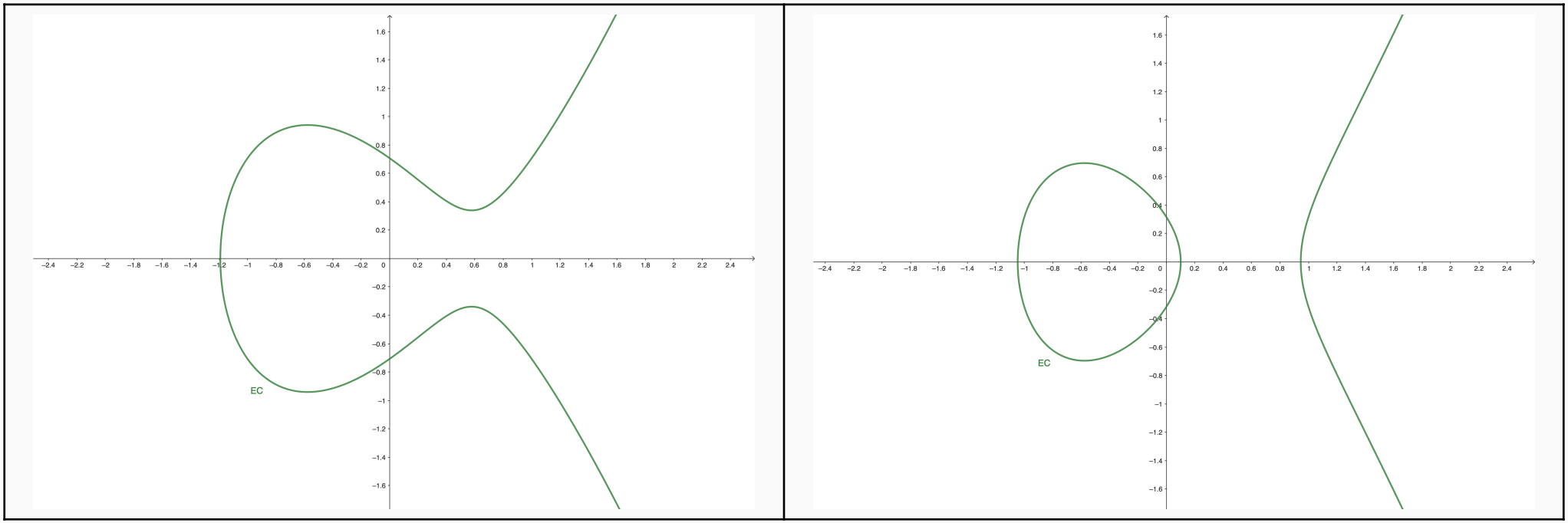
Podstawy matematyczne

Krzywe eliptyczne, używane w ECC, to zbiory punktów (x, y) spełniających równanie:

$$y^2 = x^3 + ax + b \bmod p$$

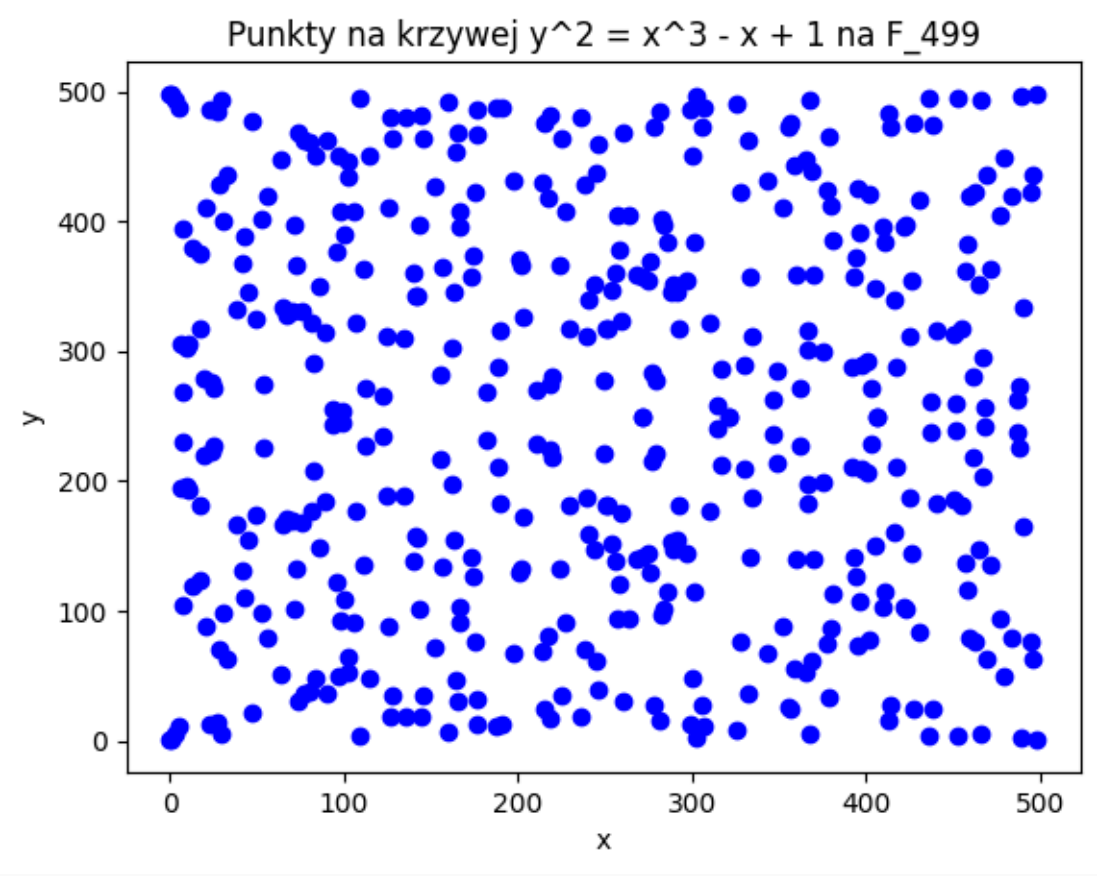
gdzie a i b są parametrami krzywej, a p jest dużą liczbą pierwszą określającą ciało skończone, nad którym operuje krzywa. Struktura ta, wraz z dodatkowym punktem nieskończoności, tworzy grupę abelową.

Krzywe eliptyczne w przestrzeni $\mathbb{R}^2$



Czy rachunki do celów kryptografii mają sens na liczbach rzeczywistych?

Krzywa eliptyczna w przestrzeni na ciele skończonym $\mathbb{F}_{499}$



Krzywe eliptyczne w openssl

```
openssl ecparam --list_curves

secp128r1 : SECG curve over a 128 bit prime field
secp128r2 : SECG curve over a 128 bit prime field
secp160k1 : SECG curve over a 160 bit prime field
...
prime192v2: X9.62 curve over a 192 bit prime field
prime192v3: X9.62 curve over a 192 bit prime field
...
sect233k1 : NIST/SECG/WTLS curve over a 233 bit binary field
sect233r1 : NIST/SECG/WTLS curve over a 233 bit binary field
sect239k1 : SECG curve over a 239 bit binary field
sect283k1 : NIST/SECG curve over a 283 bit binary field
```

Operacje na krzywych eliptycznych

Podstawowe operacje na krzywych eliptycznych to dodawanie punktów i mnożenie punktu przez skalar. Dodawanie dwóch punktów na krzywej polega na znalezieniu trzeciego punktu na krzywej, który jest ich „sumą”. Mnożenie punktu przez skalar to wielokrotne dodanie punktu do siebie samego.

$$kP = Q$$

Problem znalezienia k dla danych P i Q jest podobny do problemu znalezienia x w równaniu $a^x \equiv y \bmod n$ dla danych a i y a to jest problem logarytmu dyskretnego.

Kluczowe zastosowania

1. **Szyfrowanie**: ECC może być używane do szyfrowania danych, gdzie klucz publiczny to punkt na krzywej, a klucz prywatny to skalar.
2. **Wymiana kluczy**: Analogicznie do schematu Diffie-Hellmana, ECC może być używane do bezpiecznej wymiany kluczy, gdzie obie strony współdzielą tajny klucz na podstawie operacji na krzywych eliptycznych. Najpopularniejszym przykładem jest protokół ECDH (Elliptic Curve Diffie-Hellman).
3. **Podpisy cyfrowe**: Protokół ECDSA (Elliptic Curve Digital Signature Algorithm) pozwala na tworzenie bezpiecznych podpisów cyfrowych wykorzystujących krzywe eliptyczne.

Krzywe eliptyczne

Zalety ECC

- **Wydajność**: Krótsze klucze w ECC (np. 256-bitowe) zapewniają porównywalny poziom bezpieczeństwa do znacznie dłuższych kluczy w RSA (np. 3072-bitowe), co przekłada się na szybsze operacje i mniejsze zużycie zasobów.
- **Skalowalność**: ECC jest bardziej skalowalne niż RSA, co sprawia, że lepiej nadaje się do urządzeń o ograniczonych zasobach, jak smartfony czy urządzenia IoT.
- **Bezpieczeństwo**: Ze względu na trudność problemu logarytmu dyskretnego na krzywych eliptycznych, ECC oferuje wysoki poziom bezpieczeństwa.

Wyzwania

- **Kompleksowość implementacji**: Poprawne zaimplementowanie operacji na krzywych eliptycznych wymaga głębokiej wiedzy matematycznej i ostrożności, aby unikać błędów, które mogłyby wpłynąć na bezpieczeństwo.
- **Standardy i kompatybilność**: Wybór krzywych i parametrów musi być zgodny z obowiązującymi standardami, co jest kluczowe dla interoperacyjności i ogólnej bezpieczeństwa.

DSA - Digital Signature Algorithm

Digital Signature Algorithm (DSA) to standardowy algorytm służący do generowania i weryfikacji podpisów cyfrowych. Został wprowadzony przez National Institute of Standards and Technology (NIST) w 1991 roku jako część standardu Digital Signature Standard (DSS). DSA opiera się na trudności rozwiązania problemu logarytmu dyskretnego w grupach skończonych.

Jak działa DSA?

Parametry DSA

- p - duża liczba pierwsza (2048 lub 3072 bity)
- q - dzielnik liczby $(p - 1)$, liczba pierwsza (256 bitów)
- g - generator grupy
- x - klucz prywatny (losowa liczba z przedziału $[1, q - 1]$)
- $y = g^x \bmod p$ - klucz publiczny

Jak działa DSA?

Generowanie podpisu

1. Wybierz losową liczbę k z przedziału $[1, q-1]$
2. Oblicz $r = \left(g^k \bmod p\right) \bmod q$
3. Oblicz $s = k^{-1}(H(m) + xr) \bmod q$
4. Podpis to para (r, s)

Weryfikacja podpisu

1. Oblicz $w = s^{-1} \bmod q$
2. Oblicz $u_1 = H(m)w \bmod q$
3. Oblicz $u_2 = rw \bmod q$
4. Oblicz $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$
5. Podpis jest prawidłowy, jeśli $v = r$

Kryptografia post-kwantowa

Zagrożenie kwantowe

Komputery kwantowe stanowią poważne zagrożenie dla obecnie używanych algorytmów kryptografii asymetrycznej:

- **Algorytm Shora** (1994): może efektywnie faktoryzować duże liczby i rozwiązywać problem logarytmu dyskretnego
  - Zagrożenie dla RSA, DSA, ECDSA, Diffie-Hellman, ECDH
  - Złożoność wielomianowa: $O\big((\log n)^2(\log\log n)(\log\log\log n)\big)$
- **Algorytm Grovera** (1996): przyspiesza przeszukiwanie nieuporządkowanych zbiorów
  - Wpływ na algorytmy symetryczne (AES) i funkcje skrótu
  - Konieczność podwojenia długości kluczy (AES-128 → AES-256)

Zagrożenie kwantowe

Harmonogram zagrożenia (według NIST)

- **2030-2035**: NIST zaleca migrację do kryptografii postkwantowej; szacowane prawdopodobieństwo pojawienia się CRQC (Cryptographically Relevant Quantum Computer): 30-50% do 2030, znacznie wyższe do 2035. Najnowsze badania (Gidney 2025) wskazują, że złamanie RSA-2048 wymagałoby < 1 mln kubitów fizycznych i 1730 kubitów logicznych
- **"Harvest now, decrypt later"**: przeciwnicy mogą gromadzić zaszyfrowane dane dziś, aby odszyfrować je w przyszłości
- Długoterminowe tajemnice (np. dokumenty rządowe, medyczne) wymagają już teraz ochrony post-kwantowej

Konieczność działania

Przejście na nowe standardy wymaga czasu:

- Aktualizacja infrastruktury
- Testowanie kompatybilności
- Wymiana sprzętu i oprogramowania

Standardy NIST (2022-2024)

W lipcu 2022 roku NIST ogłosił pierwsze algorytmy post-kwantowe, które zostały zatwierdzone jako standardy:

Szyfrowanie i wymiana kluczy

CRYSTALS-Kyber (FIPS 203) - wybór główny

- Oparty na problemie Learning With Errors (LWE) na kratach algebraicznych
- Trzy poziomy bezpieczeństwa: Kyber-512, Kyber-768, Kyber-1024
- Szybki, stosunkowo małe klucze i szyfrogramy
- Przyjęty jako standard ML-KEM (Module-Lattice-Based Key Encapsulation Mechanism)

Standardy NIST (2022-2024)

Podpisy cyfrowe

CRYSTALS-Dilithium (FIPS 204) - wybór główny

- Oparty na problemie **Module Learning With Errors** (MLWE)
- Standard ML-DSA (Module-Lattice-Based Digital Signature Algorithm)
- Szybka weryfikacja, stosunkowo duże podpisy (2-4 KB)

FALCON (FIPS 206)

- Oparty na kratach NTRU
- Najmniejsze podpisy (700 bajtów), ale bardziej złożona implementacja
- Wolniejsze generowanie kluczy

SPHINCS+ (FIPS 205) - SLH-DSA

- Oparty wyłącznie na funkcjach skrótu (hash-based)
- Konserwatywna opcja (bez założeń o trudności problemów algebraicznych)
- Bardzo duże podpisy (8-50 KB), ale małe klucze

Kryptografia oparta na kratach (Lattice-based)

Kraty to regularne struktury matematyczne w przestrzeni wielowymiarowej.

Podstawowe problemy:

- SVP (Shortest Vector Problem): znalezienie najkrótszego niezerowego wektora w kracie
- CVP (Closest Vector Problem): znalezienie wektora kratowego najbliższego danemu punktowi
- LWE (Learning With Errors): odzyskanie tajnego wektora z zaszumionych równań liniowych

Zalety:

- Silne podstawy matematyczne
- Efektywność obliczeniowa
- Możliwość budowy zaawansowanych prymitywów (fully homomorphic encryption)

Kryptografia oparta na kodach (Code-based)

Oparta na trudności dekodowania losowych kodów liniowych.

Przykład: **Classic McEliece**

- Finalistka konkursu NIST (ale nie wybrana do standardu)
- Oparta na kodach Goppy
- Bardzo duże klucze publiczne (1 MB)
- Długa historia badań (od 1978 roku - kryptosystem McEliece)

Jak działa kryptografia oparta na kodach?

Podstawowa idea

Kody korekcji błędów: Metody dodawania redundancji do danych, aby możliwe było wykrycie i naprawienie błędów podczas transmisji.

Przykład: Kod powtórzeniowy

- Wiadomość: $m = 1$
- Kodowanie: $c = 111$ (powtórz 3 razy)
- Jeśli otrzymamy 101 → wiemy, że prawdopodobnie wysłano 111 (jeden błąd)

Problem dekodowania: Mając uszkodzony kod y, znajdź najbliższe poprawne słowo kodowe c.

Jak działa kryptografia oparta na kodach?

Kryptosystem McEliece - uproszczona idea

Główna idea kryptosystemu:

1. **Mamy dobry kod** (np. kod Goppy):
   - Potrafi efektywnie naprawiać błędy
   - Znamy strukturę i algorytm dekodowania
2. **Ukrywamy strukturę**:
   - Przekształcamy kod tak, aby wyglądał na losowy
   - Używamy macierzy permutacji P i transformacji S
3. **Bezpieczeństwo**:
   - Problem dekodowania losowego kodu jest trudny (NP-trudny)
   - Tylko posiadacz klucza prywatnego zna strukturę i potrafi dekodować

Klucz prywatny (sekret właściciela):

- Kod z efektywnym algorytmem dekodowania (np. kod Goppy)

Jak działa kryptografia oparta na kodach?

- Macierz generująca G tego kodu
- Macierze permutacji P i transformacji S

Klucz publiczny (dostępny dla wszystkich):

- Przekształcona macierz: $\hat{G} = SGP$
- Wygląda jak losowa macierz (ukrywa strukturę kodu)
- Każdy może szyfrować, ale tylko właściciel klucza prywatnego może odszyfrować

Jak działa kryptografia oparta na kodach?

Szyfrowanie i deszyfrowanie

Szyfrowanie wiadomości m:

1. Oblicz: $c' = m\hat{G}$ (kodowanie)
2. Dodaj losowy wektor błędów e o ustalonej wadze t
3. Szyfrogram: $c = c' + e$

Deszyfrowanie:

1. Odwróć permutację: $cP^{-1} = mSG + eP^{-1}$
2. Użyj algorytmu dekodowania kodu Goppy do naprawienia błędów
3. Odzyskaj mS, następnie $m = (mS)S^{-1}$

Bezpieczeństwo: Atakujący zna tylko $\hat{G}$ i nie wie, jak dekodować bez znajomości struktury kodu.

Kryptografia oparta na funkcjach skrótu (Hash-based)

Główna idea:

Bezpieczeństwo opiera się **wyłącznie** na odporności funkcji skrótu (np. SHA-256, SHA-3).

Jak to działa? (uproszczenie):

1. Generujemy wiele par kluczy jednorazowych (one-time signatures)
2. Organizujemy je w drzewo Merkle'a
3. Każdy podpis używa innego klucza z drzewa
4. Korzeń drzewa = klucz publiczny

Cena za bezpieczeństwo:

- Duże rozmiary podpisów (kilka-kilkadziesiąt KB)
- Wolniejsze generowanie podpisów

Kryptografia oparta na funkcjach skrótu (Hash-based)

Przykłady:

- **SPHINCS+** (wybrany przez NIST)
- **XMSS** (eXtended Merkle Signature Scheme) - dla podpisów z zachowaniem stanu

Zalety:

- Bezpieczeństwo oparte tylko na odporności funkcji skrótu
- Dobrze zrozumiane podstawy teoretyczne

Wady:

- Duże rozmiary podpisów
- W przypadku XMSS: konieczność zarządzania stanem

Co to jest izogenia?

Izogenia to specjalne odwzorowanie (funkcja) między dwiema krzywymi eliptycznymi, które zachowuje ich strukturę algebraiczną.

Intuicja geometryczna:

- Krzywa eliptyczna = zbiór punktów spełniających równanie $y^2 = x^3 + ax + b$
- Izogenia = "most" łączący dwie krzywe
- Można przejść z krzywej A do krzywej B przez izogenię

Problem kryptograficzny:

- Mając dwie krzywe eliptyczne A i B, znajdź izogenię między nimi
- Problem jest trudny nawet dla komputerów kwantowych (przypuszczalnie)

Analogia: Jak znalezienie "mapy" między dwoma labiryntami o podobnej strukturze

Kryptografia oparta na izogeniach - przykład

Przykład: **SIKE** (Supersingular Isogeny Key Encapsulation)

Historia SIKE:

- 2017-2022: Finalistka konkursu NIST (4. runda)
- **Lipiec 2022**: Złamana przez Wouter Castryck i Thomas Decru
  - Atak w czasie **jednej godziny** na pojedynczym rdzeniu procesora!
  - Wykorzystano matematyczne słabości w konstrukcji

Dlaczego to ważne?:

- Pokazuje, że nie wszystkie podejścia post-kwantowe są równie dojrzałe
- Kryptografia oparta na izogeniach wymaga dalszych badań
- Ważność kryptoanalitycznego przeglądu (peer review) w konkursach NIST

Status:

- Obecnie **nie zaleca się** używania kryptografii opartej na izogeniach w produkcji
- Badania trwają nad nowymi, bezpieczniejszymi konstrukcjami

Kryptografia oparta na izogeniach - przykład

Kryptografia oparta na równaniach wielomianowych (Multivariate)

Oparta na trudności rozwiązywania układów równań kwadratowych nad ciałami skończonymi.

Status:

- Kilka kandydatów w konkursie NIST nie przeszło do finału
- Wymaga dalszych badań

Hybrydowe podejście (zalecane)

Idea: Używać jednocześnie algorytmu klasycznego **i** post-kwantowego

Dlaczego hybrydowe?

- Algorytmy post-kwantowe są stosunkowo nowe (mniej zbadane)
- Nie wiemy na pewno, czy są bezpieczne w długim okresie
- Algorytmy klasyczne są dobrze przetestowane, ale zagrożone przez komputery kwantowe
- Kombinacja daje bezpieczeństwo w obu scenariuszach

Mechanizm: Oba algorytmy generują wspólny sekret, następnie są łączone:

```
classical_key = X25519_KeyExchange()        // np. 32 bajty
pq_key = Kyber_Encapsulate()                // np. 32 bajty
shared_secret = KDF(classical_key || pq_key)  // konkatenacja + KDF
```

`||` oznacza konkatenację (połączenie bajtów)

Przykład: **X25519 + Kyber** w TLS 1.3

Bezpieczeństwo hybrydowe:

- Jeśli X25519 zostanie złamany przez komputer kwantowy → Kyber nadal chroni
- Jeśli Kyber okaże się słaby → X25519 nadal chroni (przed klasycznym atakującym)
- Bezpieczeństwo = max(bezpieczeństwo_klasyczne, bezpieczeństwo_PQ)

Zalety:

- Bezpieczeństwo nawet jeśli jeden algorytm zostanie złamany
- Stopniowa migracja (systemy legacy mogą używać tylko klasycznych)
- Kompatybilność wsteczna

Wyzwania

1. **Rozmiary kluczy i szyfrogramów**
   - Kyber-1024: klucz publiczny 1.6 KB (vs RSA-2048: 256 bajtów)
   - Wpływ na protokoły sieciowe (TCP handshake, certyfikaty)
2. **Wydajność**
   - Algorytmy oparte na kratach są stosunkowo szybkie
   - Niektóre operacje (np. generowanie kluczy w FALCON) są wolniejsze
3. **Bezpieczeństwo implementacji**
   - Ataki side-channel (timing, power analysis)
   - Konieczność stałoczasowych implementacji
   - Trudniejsze niż w przypadku klasycznych algorytmów

Implementacja i wdrożenie

Stan wdrożeń (2024-2025)

- **Google Chrome/Chromium**: eksperymentalne wsparcie dla Kyber w TLS
- **OpenSSL 3.2+**: wsparcie dla algorytmów post-kwantowych przez provider OQS
- **Signal**: implementacja PQXDH (Post-Quantum Extended Diffie-Hellman)
- **Apple iMessage**: PQ3 protocol (2024) - ochrona post-kwantowa
- **NIST**: finalne standardy FIPS 203, 204, 205, 206 (2024)

Biblioteki i narzędzia

- **liboqs** (Open Quantum Safe): implementacje algorytmów post-kwantowych
- **PQClean**: zbiór czystych, przenośnych implementacji w C
- **Bouncy Castle**: wsparcie w Java i C#

Przyszłość kryptografii post-kwantowej

Kolejne rundy NIST

- **Dodatkowe algorytmy**: NIST kontynuuje ocenę kolejnych kandydatów
- Potrzeba:
  - Algorytmów z różnymi trade-offami (rozmiar vs. szybkość)
  - Alternatywnych podejść matematycznych (dywersyfikacja ryzyka)

Kwantowa dystrybucja klucza (QKD)

- **Quantum Key Distribution**: wykorzystanie zjawisk kwantowych do dystrybucji kluczy
- Bezpieczeństwo oparte na prawach fizyki (niemożność sklonowania stanu kwantowego)
- Ograniczenia: zasięg, koszty infrastruktury, wymaga dedykowanych połączeń

Rekomendacje

Dla organizacji

1. **Inwentaryzacja**: zidentyfikować systemy używające kryptografii asymetrycznej
2. **Priorytetyzacja**: długoterminowe tajemnice wymagają natychmiastowej ochrony
3. **Testowanie**: rozpocząć testy z algorytmami post-kwantowymi w środowiskach testowych
4. **Kryptoagility**: projektować systemy umożliwiające łatwą wymianę algorytmów

Dla programistów

- Używać bibliotek zgodnych ze standardami NIST
- Implementować hybrydowe rozwiązania
- Monitorować rozwój standardów i aktualizacje bezpieczeństwa
- Unikać samodzielnych implementacji niskopoziomowych

Podsumowanie

- **Zagrożenie kwantowe jest realne** i wymaga działania już teraz
- **Standardy NIST** dostarczają sprawdzonych algorytmów
- **Hybrydowe podejście** zapewnia bezpieczeństwo w okresie przejściowym
- **Kryptografia oparta na kratach** (Kyber, Dilithium) to główny kierunek
- **Wdrożenie wymaga czasu** - należy rozpocząć przygotowania

Bezpieczeństwo kwantowe nie jest kwestią “czy”, ale “kiedy”