Kryptografia na Usługach Dewelopera

Cezary Kujawa

Cel:

- Uporządkowanie i pogłębienie wiedzy
- Zainteresowanie fascynującą dziedziną

Kryptologia

(z gr. κρυπτός – **kryptos** – „ukryty” i λόγος – **logos** – „rozum”, „słowo”) – dziedzina wiedzy o przekazywaniu informacji w sposób zabezpieczony przed niepowołanym dostępem. Współcześnie kryptologia jest uznawana za gałąź zarówno matematyki, jak i informatyki; ponadto jest blisko związana z teorią informacji, inżynierią oraz bezpieczeństwem komputerowym.

źródło: wikipedia

Kryptologia

Kryptografia

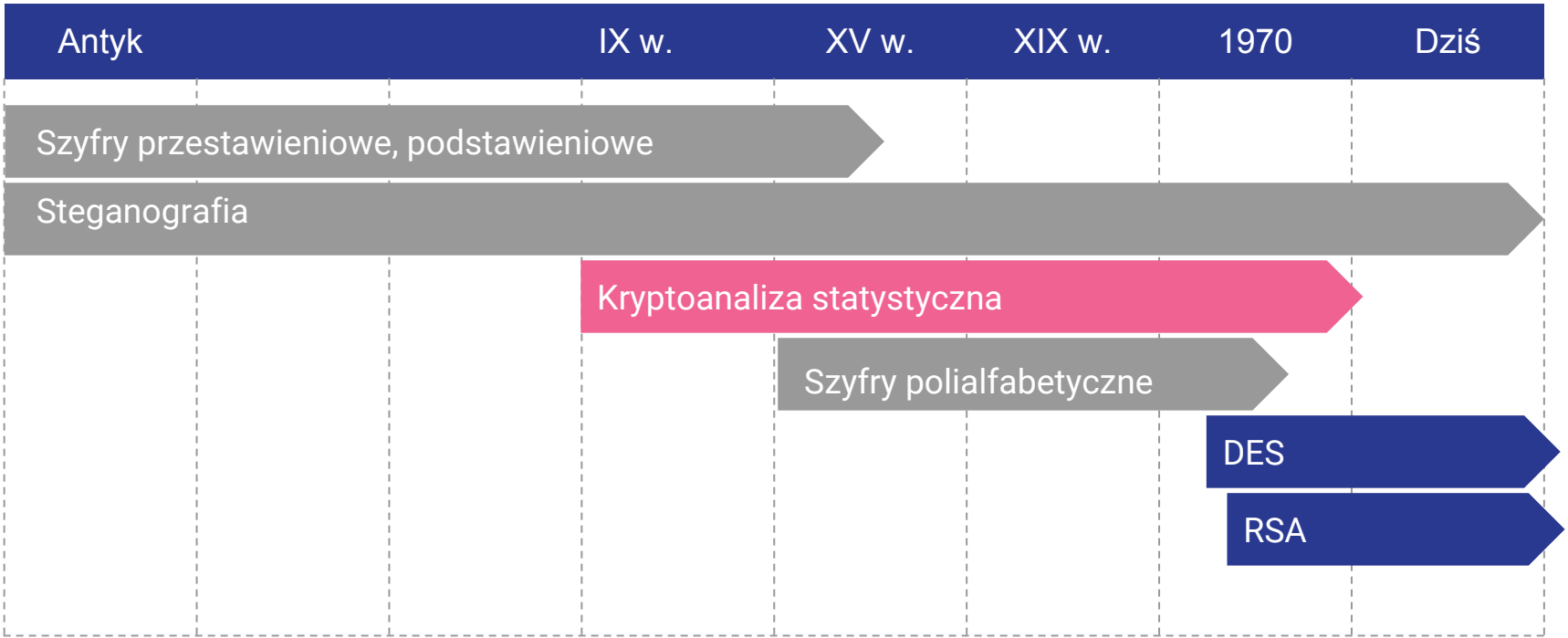
(z gr. κρυπτός – **kryptos** – „ukryty” oraz γράφω **gráfo** „pisać”), czyli gałąź wiedzy o utajnianiu wiadomości; a także: praktyczne wykorzystanie technik z dziedziny nauki – kryptologii.

Kryptoanaliza

(z gr. κρυπτός – **kryptos** – „ukryty” oraz **analýein** – rozluźnić), czyli gałąź wiedzy o przełamywaniu zabezpieczeń oraz o deszyfrowaniu wiadomości przy braku klucza lub innego wymaganego elementu schematu szyfrowania (szyfru).

źródło: wikipedia

Historia



źródło: wikipedia

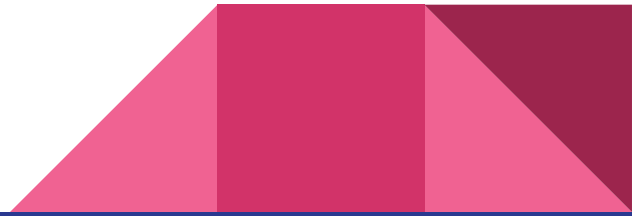
Funkcje Kryptografii

Bez klucza	Symetryczna	Asymetryczna
• **funkcje skrótu** (hash functions)	• **skróty haseł** (password hashing) • **uwierzytelnianie** (authentication) • **szyfrowanie** (encryption)	• **uwierzytelnianie** (authentication) • **szyfrowanie** (encryption) • **wymiana kluczy** (key exchange)

źródło: PARAGON INITIATIVE

Funkcja skrótu

Algorytm matematyczny, który na podstawie wiadomości dowolnego rozmiaru tworzy ciąg bajtów o ustalonej wielkości.



Cechy funkcji skrótu

Jednokierunkowa

Brak możliwości odtworzenia wiadomości na podstawie skrótu.

Deterministyczna

Identyczny skrót oblicznay dla tej samej wiadomości

Szybka

Niska złożoność obliczeniowa względem długości wiadomości

Zmiennowartościowa

Mała zmiana w wiadomości powoduje dużą zmienność wartości skrótu

Odporność na kolizje

O wartości decyduje złożoność obliczeniowa operacji pozwalającej na stworzenie drugiej wiadomości o tej samej wartości skrótu (kolizji)

Zastosowania funkcji skrótu

Sumy kontrolne

Pozwala sprawdzić czy dane są identyczne:

- weryfikacja integralności
- eliminacja błędów

Dostęp do danych

Optymalizacja dostępu do danych na podstawie ich wartości.

- HashMap, HashSet
- Identyfikator pliku, rewizji w

Podstawa innych

Używana wraz z innymi algorytmami w celu:

- weryfikacji hasła
- podpis wiadomości
- HMAC

Zalecenia

Amerykański instytut NIST publikuje zalecenia dotyczące stosowania poszczególnych funkcji skrótu w zależności od pożądanego czasu ochrony informacji.

Od 1999	Nie powinna być stostowana funkcja MD5
Od 2010	Nie powinna być stotsowana funkcja SHA-1
Dziś	Do nowych aplikacji zalecane są funkcje skrótu z rodziny SHA-2, a w przyszłości funkcja SHA-3

źródło: nist.gov

Przykłady funkcji skrótu

MD5

Message-Digest Algorithm 5

- opracowany w 1991 (Ron Rivest)
- 128-bitowy skrót
- w 2004 znaleziono sposób na generowanie kolizji

SHA

Secure Hash Algorithm

- opracowany przez NSA
- opublikowany przez NIST
- 1993 SHA-0
- 1995 SHA-1
- 2001 SHA-2
- 2012 SHA-3

BLAKE2

BLAKE Hash Function

- szybszy niż MD5 i SHA
- bezpieczny jak SHA-3
- RFC 7693
- opracowany w 2015

Przykłady

Java

```
String message = "Ala ma kota";
MessageDigest digest = MessageDigest.getInstance("SHA-256");
byte[] hash = digest.digest(message.getBytes("UTF8"));
System.out.println(Hex.encodeHexString(hash));

>> 124bfb6284d82f3b1105f88e3e7a0ee02d0e525193413c05b75041917022cd6e
```

OpenSSL

```
$ echo -n "Ala ma kota" | openssl dgst -sha256
(stdin)= 124bfb6284d82f3b1105f88e3e7a0ee02d0e525193413c05b75041917022cd6e
```

Proof of work

To fragment danych, który jest trudny (kosztowny) do wytworzenia, ale łatwy do weryfikacji. Jednym ze sposobów realizacji może być losowy proces o niskim prawdopodobieństwie powodzenia.

HashCash

Bitcoin wykorzystuje hashcash-SHA256^2 w procesie wydobywania waluty.

Może być wykorzystywany do walki ze spamem.

źródło: bitcoin.it

Funkcja skrótu - częste błędy

Przechowywanie haseł

Używanie szybkiej funkcji skrótu jako jedynego zabezpieczenia przechowywanych haseł

- podatne na brute force
- należy użyć **bcrypt** lub **PBKDF2**

Skracanie identyfikatorów

Wykonywanie funkcji skrótu na identyfikatorach w celu ich ukrycia

- podatne na brute force

Źródło: PARAGON INITIATIVE/An Open Letter to Developers Everywhere (About Cryptography)

Funkcje Kryptografii

Bez klucza	Symetryczna	Asymetryczna
• **funkcje skrótu** (hash functions)	• **skróty haseł** (password hashing) • **uwierzytelnianie** (authentication) • **szyfrowanie** (encryption)	• **uwierzytelnianie** (authentication) • **szyfrowanie** (encryption) • **wymiana kluczy** (key exchange)

źródło: PARAGON INITIATIVE

Cechy kryptografii symetrycznej

Tajny klucz

Parametrem algorytmu są dane oraz tajny klucz

Wspólny sekret

Klucz do odszyfrowania wiadomości identyczny lub wymagający prostej transformacji

Dwustronna

Wynikiem szyfrowania jest kryptogram, z którego można odtworzyć wiadomość

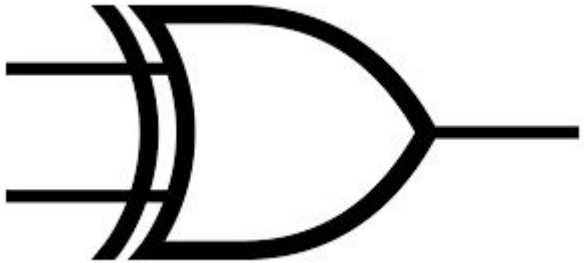
Wymaga bezpiecznego kanału

Klucz powinien być przekazany tzw. “bezpiecznym kanałem”

Szyfr z kluczem jednorazowym

XOR

$$0 \oplus 0 = 0$$
$$0 \oplus 1 = 1$$
$$1 \oplus 0 = 1$$
$$1 \oplus 1 = 0$$



Szyfr z kluczem jednorazowym

- niemożliwy do złamania
- każdy bit klucza wykorzystany tylko raz
- długość klucza = długości tekstu jawnego

Podział algorytmów symetrycznych

Strumieniowe

Szyfrują kolejne słowa (zazwyczaj bajty) komunikatu, dowolna długość klucza, wynik szyfrowania kolejnych słów zależny od stanu maszyny szyfrującej

Przykłady:

- RC4 (używany w protokołach SSL i WEP)
- Salsa20
- Chacha20

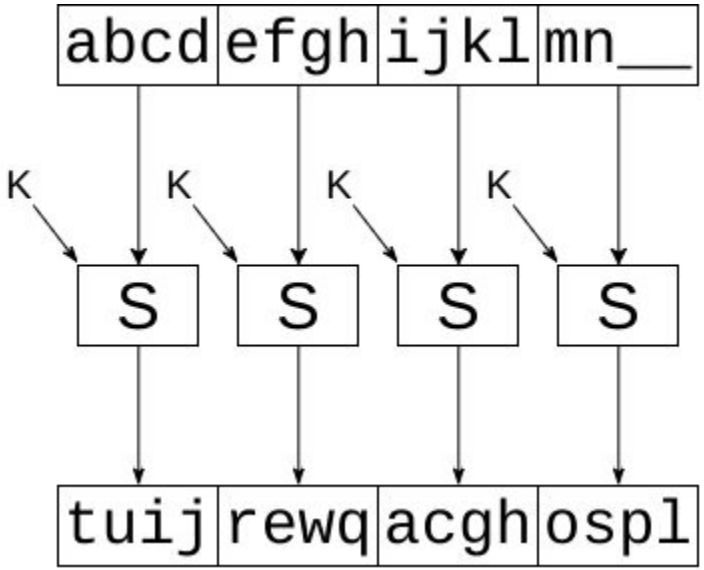
Blokowe

Szyfrują wielobitowe bloki komunikatu, dopełniają (padding) wiadomość do pełnej wielokrotności bloku, z góry zdefiniowane długości klucza.

Przykłady:

- DES (Data Encryption Standard)
- Blowfish
- AES (Advanced Encryption Standard)
- IDEA

Szyfrowanie blokowe



Dopełnienie

Dopełniania PKCS#5 i PKCS#7

Każdy bajt dopełnienia ma wartość ilości bajtów

```
01
02 02
03 03 03
04 04 04 04
05 05 05 05 05
06 06 06 06 06 06
```

Tryby szyfrów blokowych

ECB electronic codebook

każdy blok szyfrowany oddzielnie, ułatwia wielowątkowe przetwarzanie

PCBC plaintext cipher-block chaining

Każdy kolejny blok dodawany XOR do poprzedniego bloku jawnego i szyfrogramu

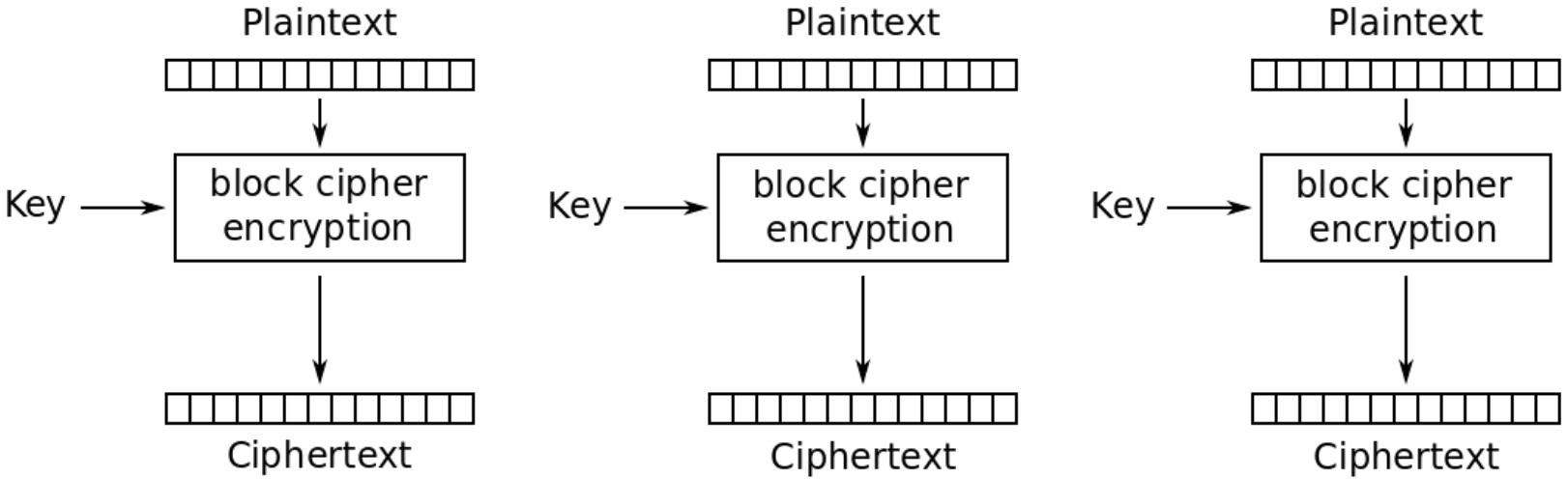
CBC cipher block chaining

Każdy kolejny blok dodawany XOR do poprzedniego bloku szyfrogramu

CFB cipher feedback

Szyfrowaniu podlega nie tekst jawny ale po zsumowaniu XOR z poprzednim blokiem szyfrogramu

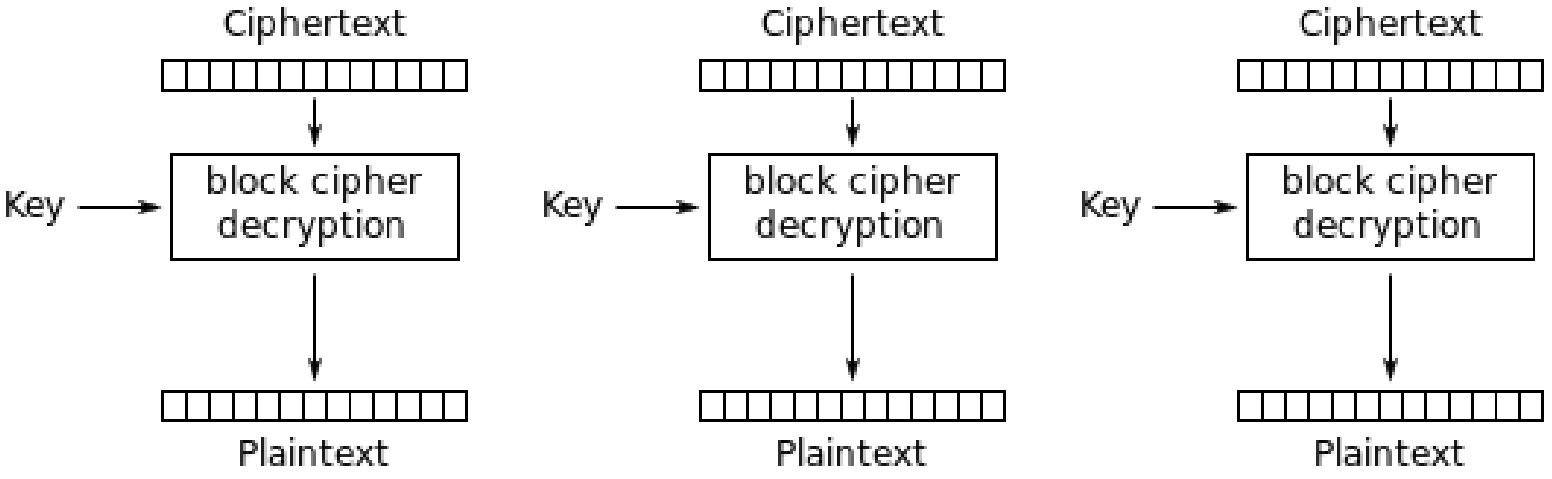
Tryb ECB szyfrowanie



Electronic Codebook (ECB) mode encryption

źródło: wikipedia

Tryb ECB deszyfrowanie

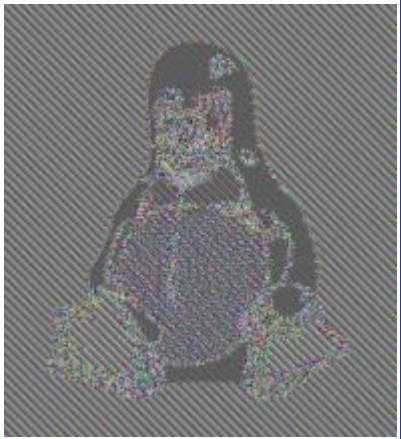


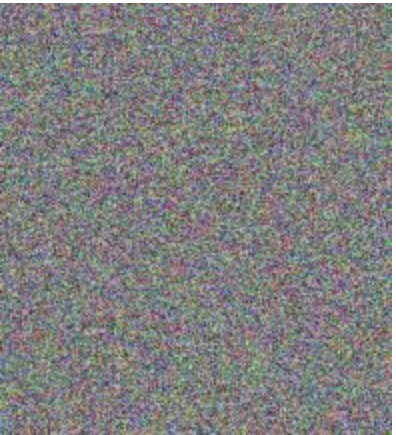
Electronic Codebook (ECB) mode decryption

źródło: wikipedia

komunikat

szyfrogram ECB





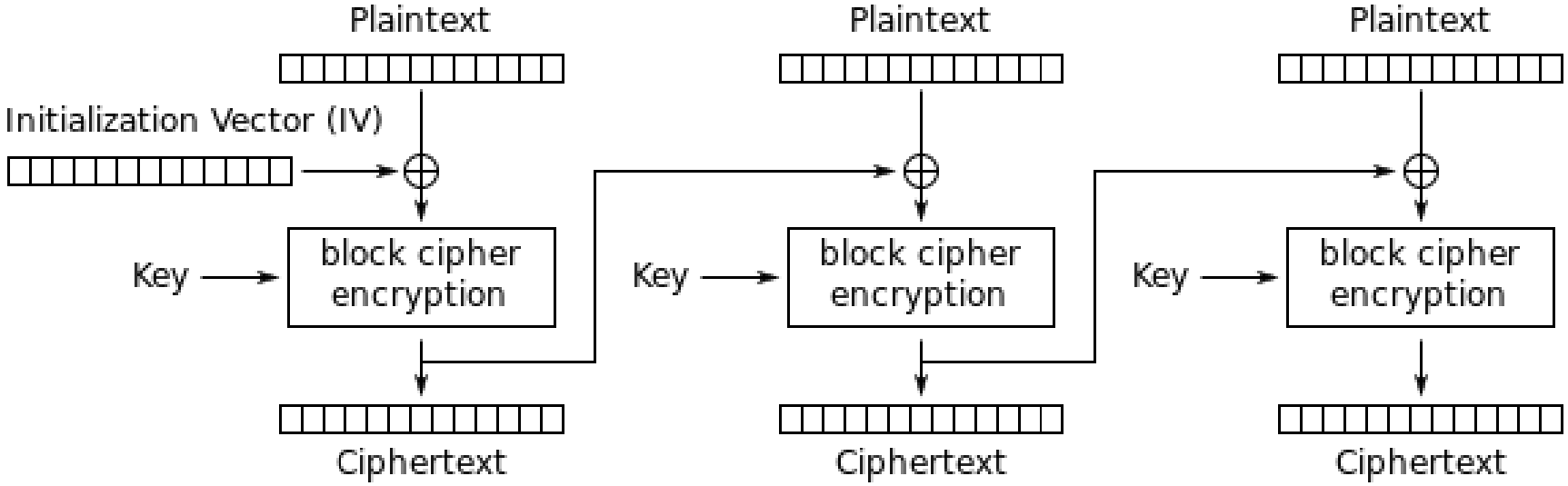
szyfrogram CBC

Tryb ECB

- każdy blok szyfrowany oddzielnie
- ułatwia wielowątkowość
- bloki równe w komunikacie równe w szyfrogramie
- podatny na kryptonalizę statystyczną

Oryginalny rysunek stworzony przez użytkownika: Lunkwill anglojęzycznej wikipedii w 2004 roku

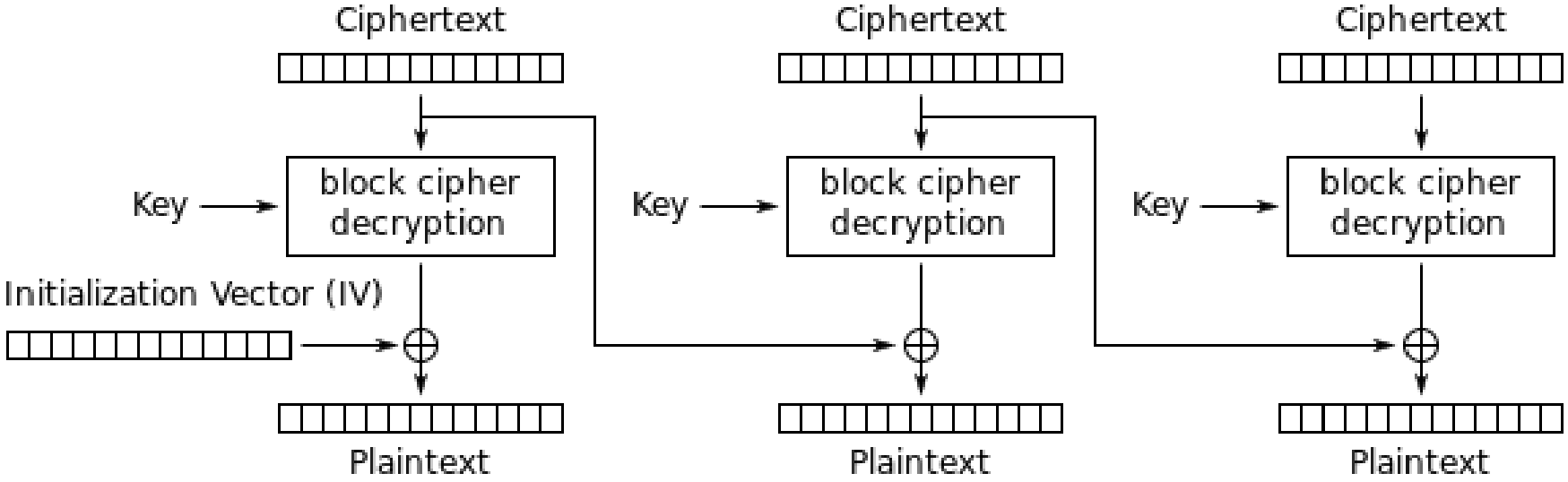
CBC szyfrowanie



Cipher Block Chaining (CBC) mode encryption

źródło: wikipedia

CBC deszyfrowanie



Cipher Block Chaining (CBC) mode decryption

źródło: wikipedia

Przykłady

Java / `javax.crypto`

```
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
SecretKeySpec key = new SecretKeySpec("changeItchangeIt".getBytes(), "AES");
cipher.init(Cipher.ENCRYPT_MODE, key);
byte[] encrypted = cipher.doFinal("Ala ma kota".getBytes());
System.out.println(Hex.encode(encrypted));
cipher.init(Cipher.DECRYPT_MODE, key);
byte[] decrypted = cipher.doFinal(encrypted);
System.out.println(new String(decrypted));

>> c531dedabea74ad00151e6a1db9b2b3f
>> Ala ma kota
```

Przechowywanie haseł

Złe praktyki

Funkcja skrótu

Zaletą funkcji skrótu jest niski koszt obliczeniowy co powoduje, że jest podatna na atak brute force. Nowoczesne karty graficzne potrafią wykonać miliardy skrótów na sekundę

Sól

losowa wartość dołączana do hasła w celu zróżnicowania wyników skrótu zabezpiecza przed atakami z tęczową tablicą (rainbow table) jednak nie zabezpiecza przed brute force.

Przechowywanie haseł

Password-Based Key Derivation Functionions

bcrypt

- **kosztowna** bazuje na kosztownym algorytmie generowania klucza z blowfish
- **adaptacyjna** parametr decyduje o koszcie

PBKDF2

- **kosztowna** wykonuje funkcję pseudolosową, jaką jest HMAC
- **adaptacyjna** Ilość iteracji pozwala spowolnić algorytm

script

- **kosztowna** poza kosztem obliczeniowym nakłada też koszt związany wykorzystywaną pamięcią

Przykłady

Java / Spring Security

```
PasswordEncoder passwordEncoder = new BCryptPasswordEncoder(10);
          for (int i = 0; i < 2; i++) {
               String encoded = passwordEncoder.encode("changeIt");
               System.out.println(encoded);
          }
```

```
>> $2a$10$ervF8kX7SGUBXiPemLpkSuetPQ.D3YY2ViIHUmwZzajoTVc13YGhq
>> $2a$10$ZHiRIk8RsK7bRvnDIpHVhOZCQ.RTM/uZVwh/QqBxJgU59nlxnd8Li
```

$2a$10 oznacza wartość parametru kosztu 2^10
ervF8kX7SGUBXiPemLpkSu wygenerowana sól
etPQ.D3YY2ViIHUmwZzajoTVc13YGhqv wygenerowany skrót hasła

Uwierzytelnienie + Integralność = Podpis

MAC

Message Authentication Code - informacja potwierdzająca pochodzenie oraz integralność komunikatu.

- komunikat jawny
- wyposażony w tag (pieczęć) wygenerowany z wykorzystaniem klucza

HMAC

Hash-based Message Authentication Code

- wykorzystuje funkcję skrótu
- HMAC-MD5
- HMAC-SHA1
- HMAC-SHA256

Przykłady

Java / javax.crypto

```
Mac hmac = Mac.getInstance("HmacSHA256");
hmac.init(new SecretKeySpec("changeIt".getBytes(), "HmacSHA256"));
byte[] signature = hmac.doFinal("Ala ma kota".getBytes());
System.out.println(Hex.encode(signature));
```

```
>> 87f76d103fc6c74b7bb558d6e144368fd20b810548023455d0176bb348a40af8
```

Kryptografia symetryczna - błędy

Kryptogram jako uwierzytelnienie

Kryptogram nie jest tajny, posiadacz nie musi być autorem.

MAC przed szyfrowaniem

Operacje MAC powinny być zawsze wykonywane jako ostatnie.

Przewidywalny wektor inicjujący CBC

Wektor inicjujący zawsze powinien być wartością losową

Czytelne hasło kluczem

Kluczem w szyfrowaniu jest ciąg bajtów, użycie czytelnego hasła drastycznie zmniejsza siłę klucza

Funkcje Kryptografii

Bez klucza	Symetryczna	Asymetryczna
• **funkcje skrótu** (hash functions)	• **skróty haseł** (password hashing) • **uwierzytelnianie** (authentication) • **szyfrowanie** (encryption)	• **uwierzytelnianie** (authentication) • **szyfrowanie** (encryption) • **wymiana kluczy** (key exchange)

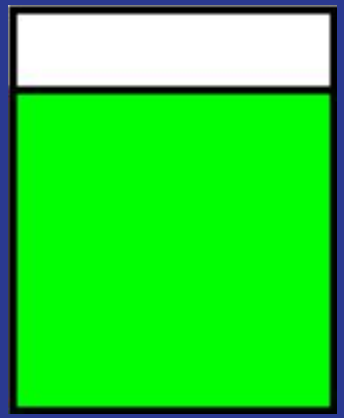
źródło: PARAGON INITIATIVE

Protokół Diffiego-Hellmana

protokół uzgadniania kluczy szyfrujących, opracowany przez Witfielda Diffiego oraz Martina Hellmana w 1976 roku. Jego siła oparta jest na trudności obliczenia logarytmów dyskretnych w ciałach skończonych.

źródło: wikipedia

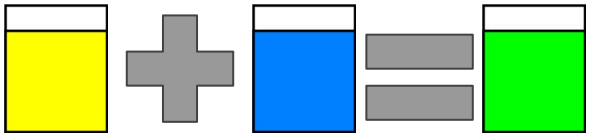
Farbki



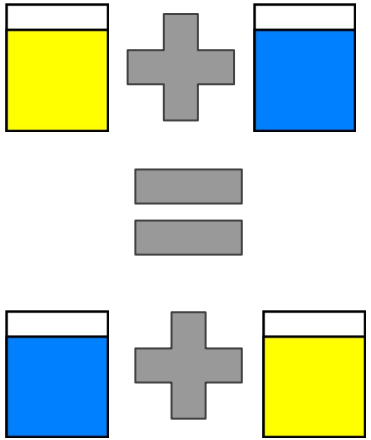
źródło: crypto101.io

Zasady dotyczące farbek

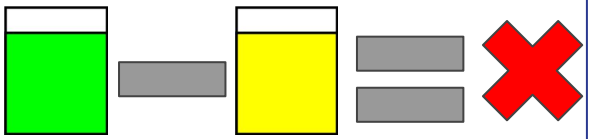
Mieszanie



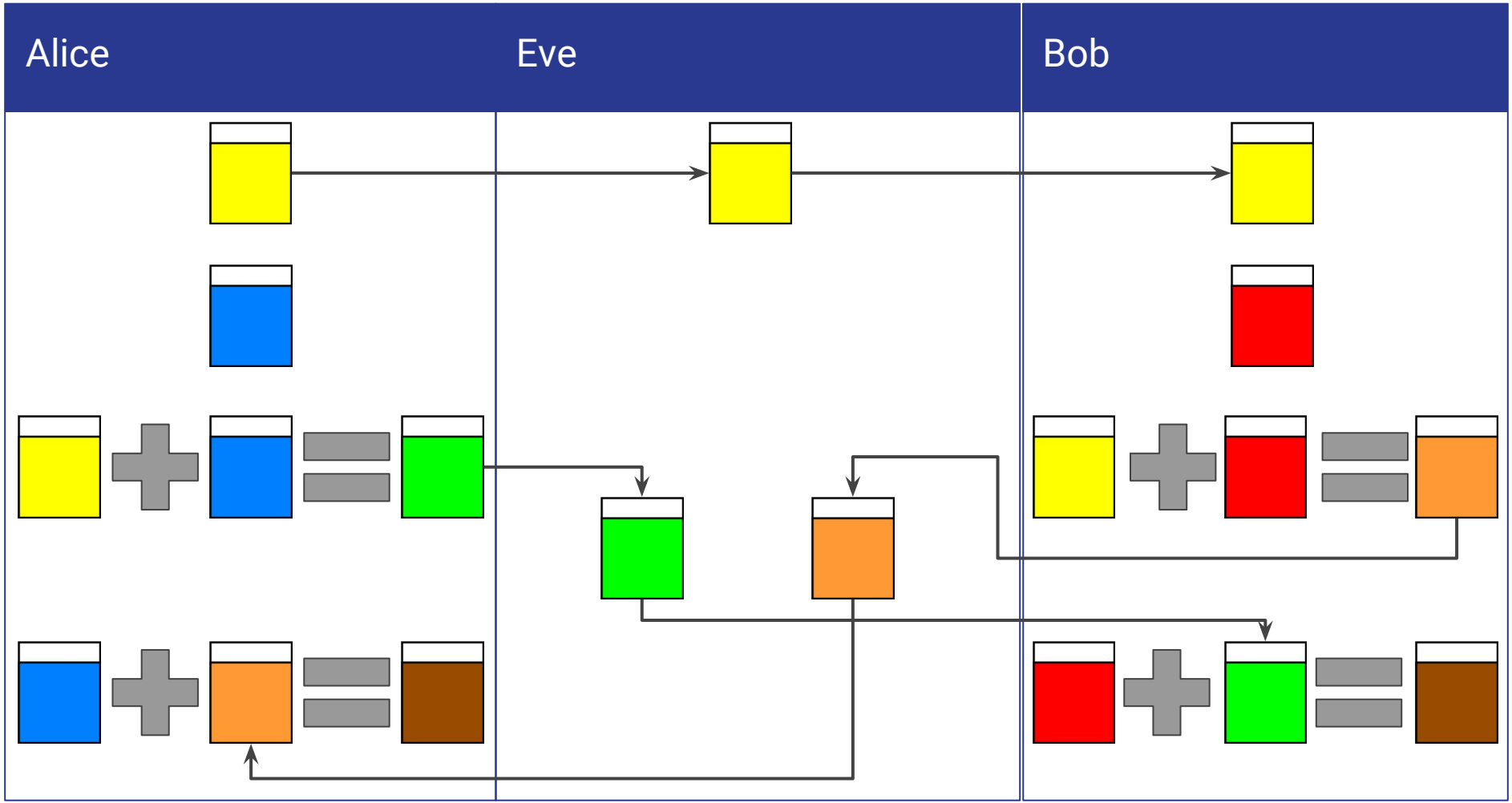
Kolejność nieistotna



Nie można rozdzielić

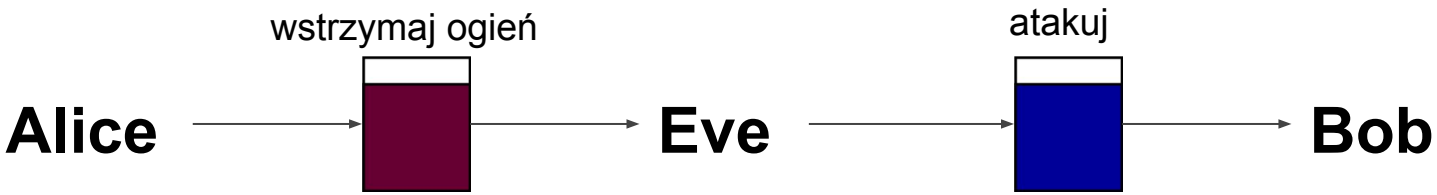


źródło: crypto101.io



źródło: crypto101.io

Problem: **Podszywanie**



źródło: crypto101.io

Rozwiązanie:
kryptografia klucza publicznego

Cechy kryptografii asymetrycznej

Para kluczy

Publiczny klucz rozpowszechniony, klucz prywatny chroniony

Nie wymaga bezpiecznego kanału

Nie ma potrzeby przekazywania klucza prywatnego

Kosztowna

Wysoka złożoność obliczeniowa powoduje, że w praktyce używa się tylko do szyfrowania niewielkich ilości danych np. do zaszyfrowania klucza symetrycznego, który może potem być wykorzystany do dalszego szyfrowania.

Rivest Shamir Adleman

Jeden z pierwszych, najpopularniejszy kryptosystem klucza publicznego

- zaprojektowany w 1977 przez Rona Rivesta, Adi Szamira oraz Leonarda Adlemana
- bezpieczeństwo opiera się na trudności faktoryzacji dużych liczb złożonych
- zastosowania:
  - szyfrowanie
  - podpisy cyfrowe

Klucze

Generowanie kluczy

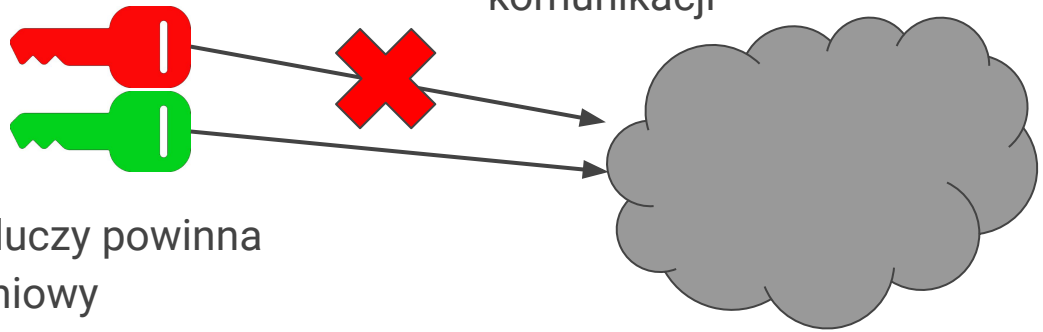
Efektem jest para matematycznie spokrewnionych kluczy:

- klucz prywatny
- klucz publiczny

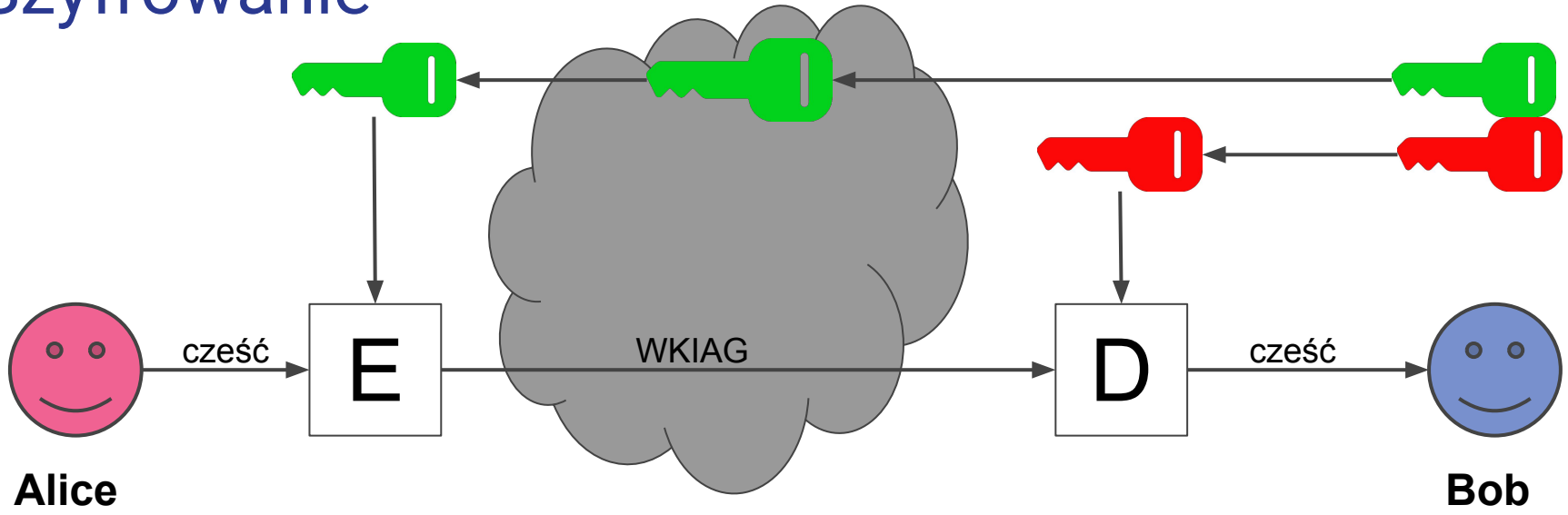
Operacja generowania kluczy powinna mieć niski koszt obliczeniowy

Dystrybucja klucza publicznego

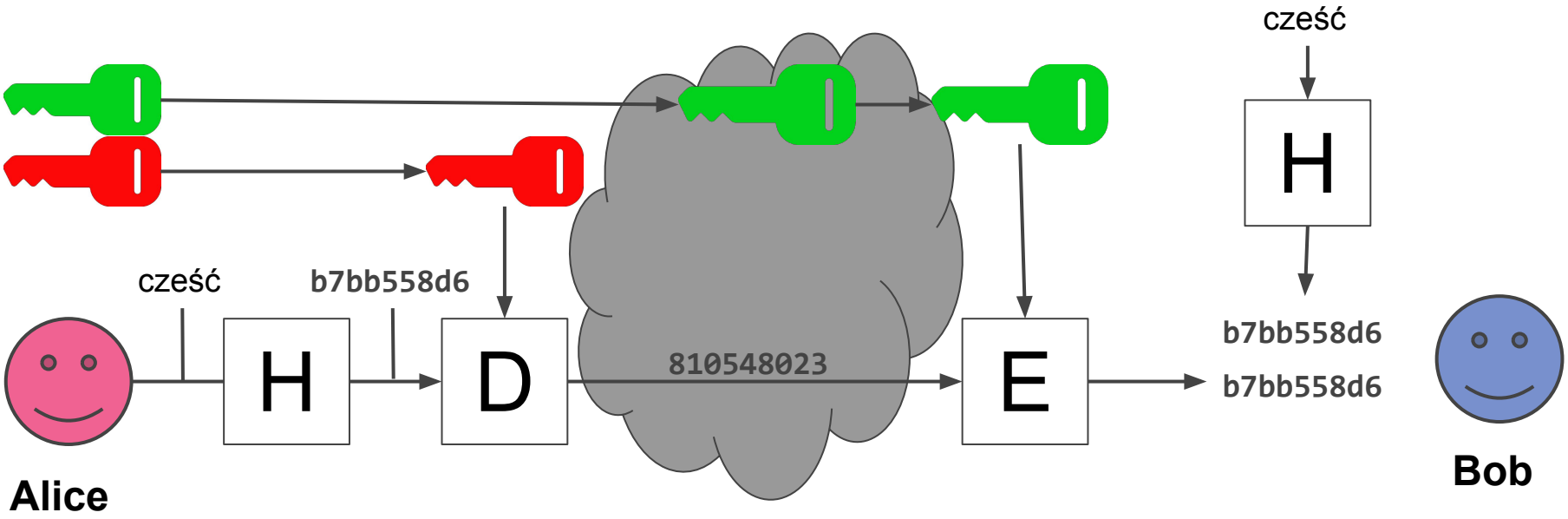
Klucz publiczny jest jawny i dystrybuowany w celu umożliwienia komunikacji



Szyfrowanie



Podpisywanie



Certyfikat

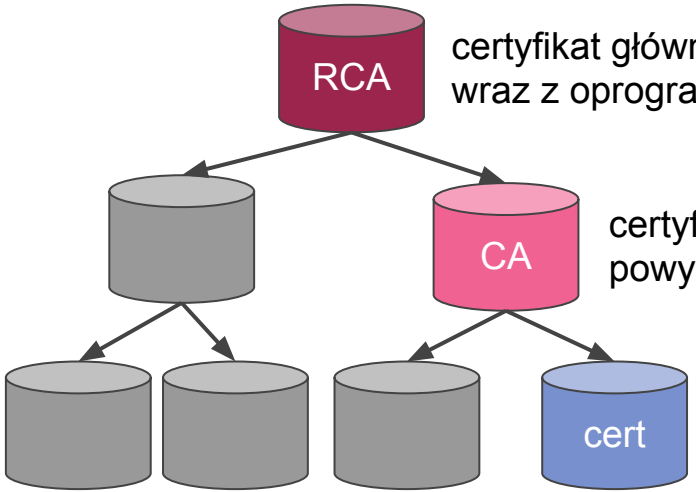
Potwierdza własność klucza publicznego

Składa się z:

- klucza publicznego
- podpisu złożonego przez zaufany urząd certyfikacji (CA)

Łańcuch certyfikacji

- VeriSign Class 3 Public Primary Certification Authority - G5
  - Symantec Class 3 ECC 256 bit EV CA - G2
    - www.java.com



certyfikat głównego urzędu certyfikacji (Root CA) rozpowszechniany wraz z oprogramowaniem

certyfikat pośredniego urzędu certyfikacji potwierdzony przez powyższy certyfikat

certyfikat potwierdzający własność klucza publicznego do komunikacji ze stroną java.com

Biblioteki

- Spring Security
  - uwierzytelnienie
  - przechowywanie hasła
- Bouncy Castle
  - API i implementacja wielu funkcji i algorytmów
- google/keyczar
  - bardzo proste API
  - utrudnia niewłaściwe praktyki
- abstractj/kalium
  - Interfejs w Javie do natywnych bibliotek NaCl

Dziękuję bardzo!

Cezary Kujawa

cezary.kujawa@globallogic.com