

\\
.
001.
^
u\$ON=1
z00BAI
I...=..
;s<...
NRX~=-
z0c^<X^
~B0s~^
00\$H~
n\$0=XN;..
iBB0vU1=~..
`\$000cRr`vul
FAHZuqr-
ZZUFA0FI..
;BRHv n\$U^
`ARN1 ^0si
'Onv~ 01..
c0qr rs..
aUU` ul..
`RO- :..
nn~` -=.~|-
=1^'...` :..

Podstawowe elementy kryptografii

część II

Zagadnienia

1. Terminologia
2. Szyfry symetryczne
3. Szyfry asymetryczne
4. Funkcje skrótu i podpis cyfrowy
5. Praktyczne problemy kryptografii
6. Zastosowania kryptografii
7. Przyszłość kryptografii
8. Steganografia

Autentyczność

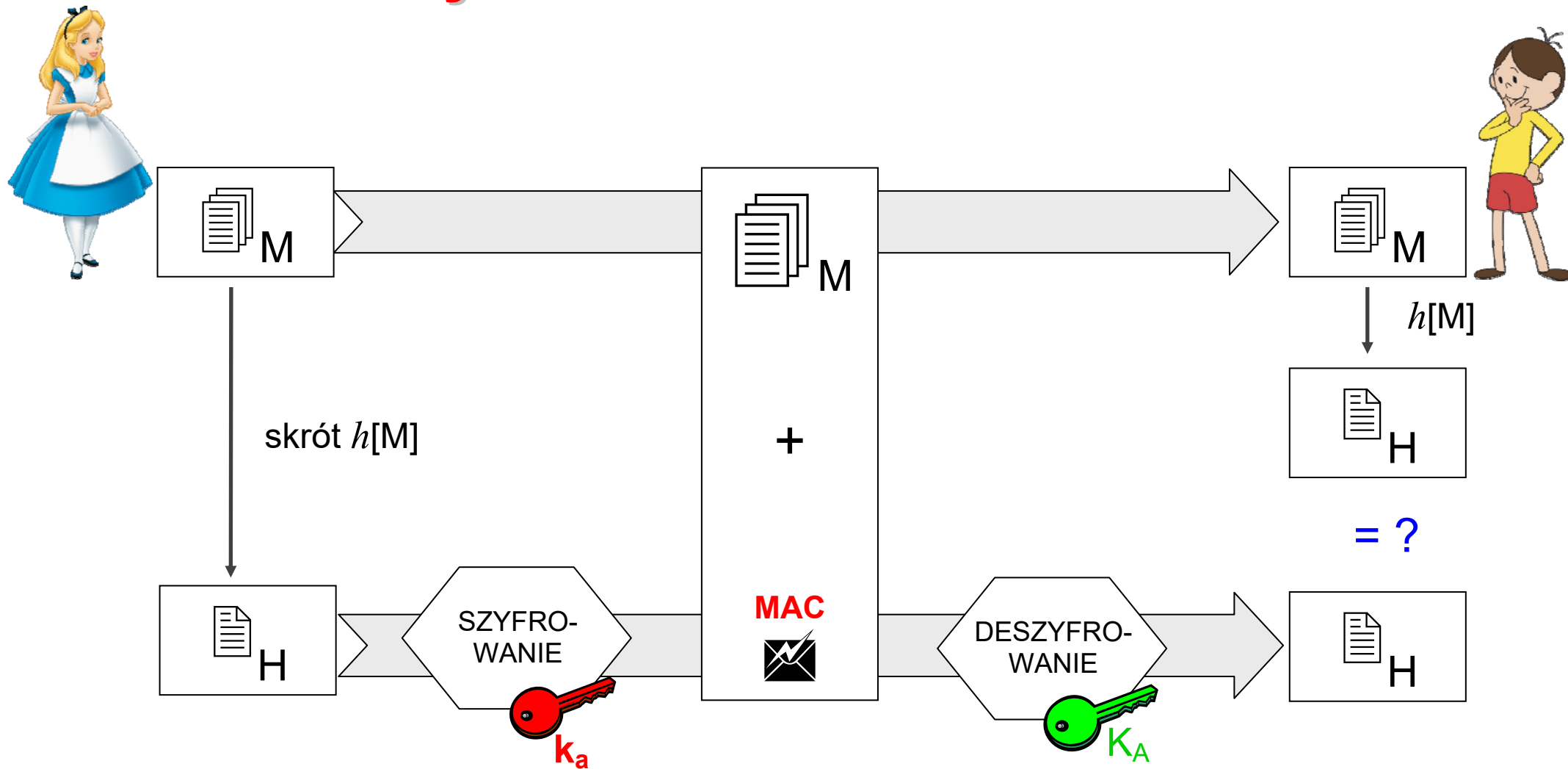
Metoda 1:

- szyfrujemy całą wiadomość
- kosztowne obliczeniowo – koszt rośnie z wielkością wiadomości

Metoda 2:

- tworzymy skrót wiadomości o ustalonym z góry rozmiarze n
- szyfrujemy tylko skrót
- koszt mały – n małe
- koszt stały – nie rośnie z wielkością wiadomości i zależy tylko od n

Autentyczność



MAC – *message authentication code*

Autentyczność

Funkcja skrótu

- jednokierunkowa funkcja $h[M]$
- $H=h[M]$ – skrót o stałym rozmiarze n
- najczęściej – funkcja mieszająca (*hash function*)
- być może z ziarnem (*salt*) lub zawołaniem (*challenge*)

+ szyfrowanie

- asymetryczne: MAC – *message authentication code*
- symetryczne: ICV – *integrity check value*

Skrót

Rys historyczny:

skrót w informatyce powszechnie wykorzystywane są od lat '70

- suma kontrolna (*checksum*)
- funkcja kontrolna danych (*data integrity check*)
- funkcja ściągająca (*contraction function*)
- odcisk palca (*fingerprint*) danych
- skrót wiadomości (*message digest*)
- ...

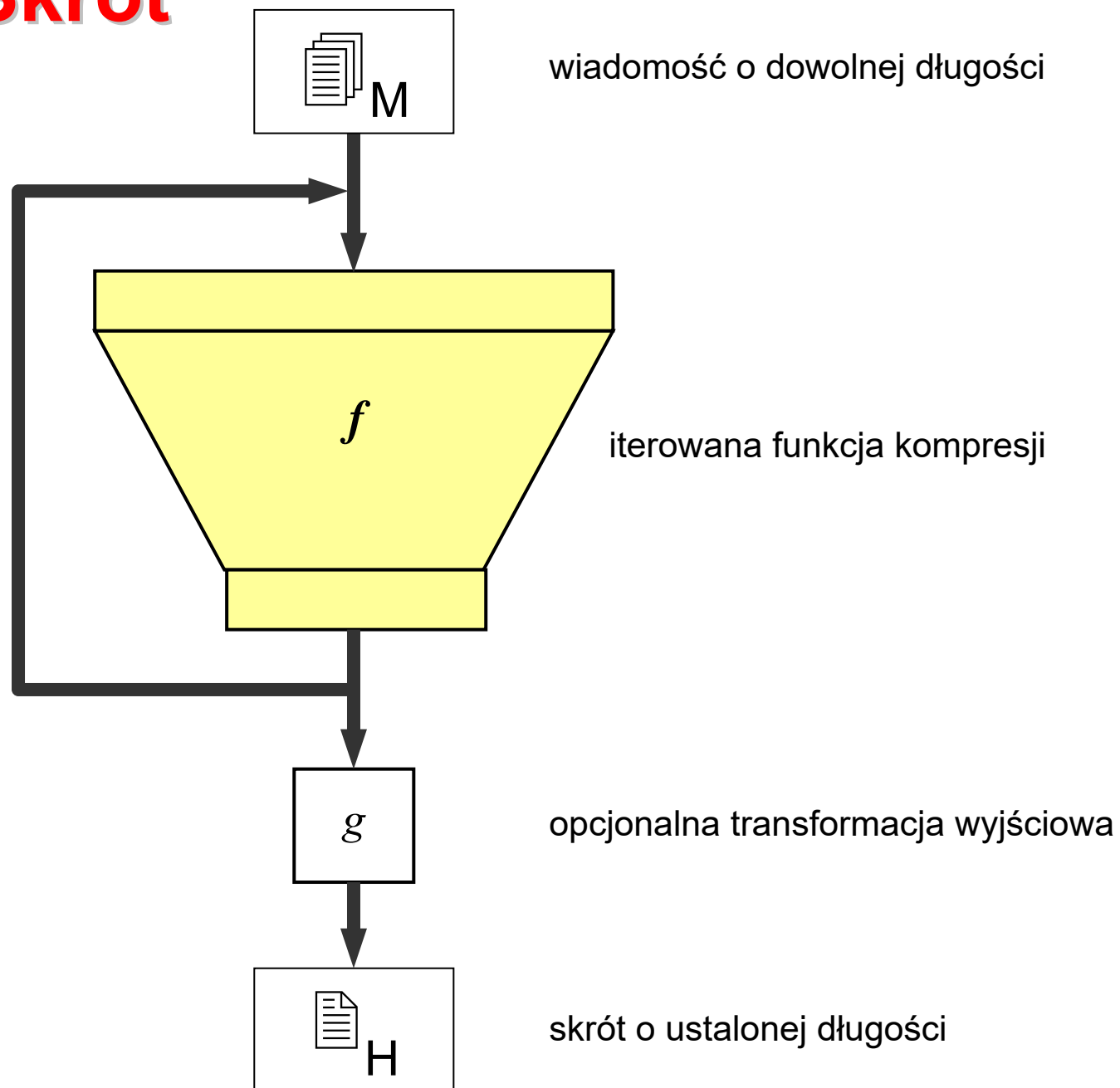
Skrót

Wymagane własności:

- kompresja: $|H| < |M|$
- łatwość obliczeń: czas wielomianowy wyznaczenia $h[M]$ dla dowolnego M
- odporność na **podmianę** argumentu: dla danego $h[M]$ obliczeniowo trudne znalezienie $M' :: h[M] = h[M']$
- odporność na **kolizje**: obliczeniowo trudne znalezienie dwóch dowolnych argumentów $M \neq M' :: h[M] = h[M']$
- możliwie losowy rozkład wartości: prawdopodobieństwo podmiany/kolizji zależne jedynie od długości skrótu

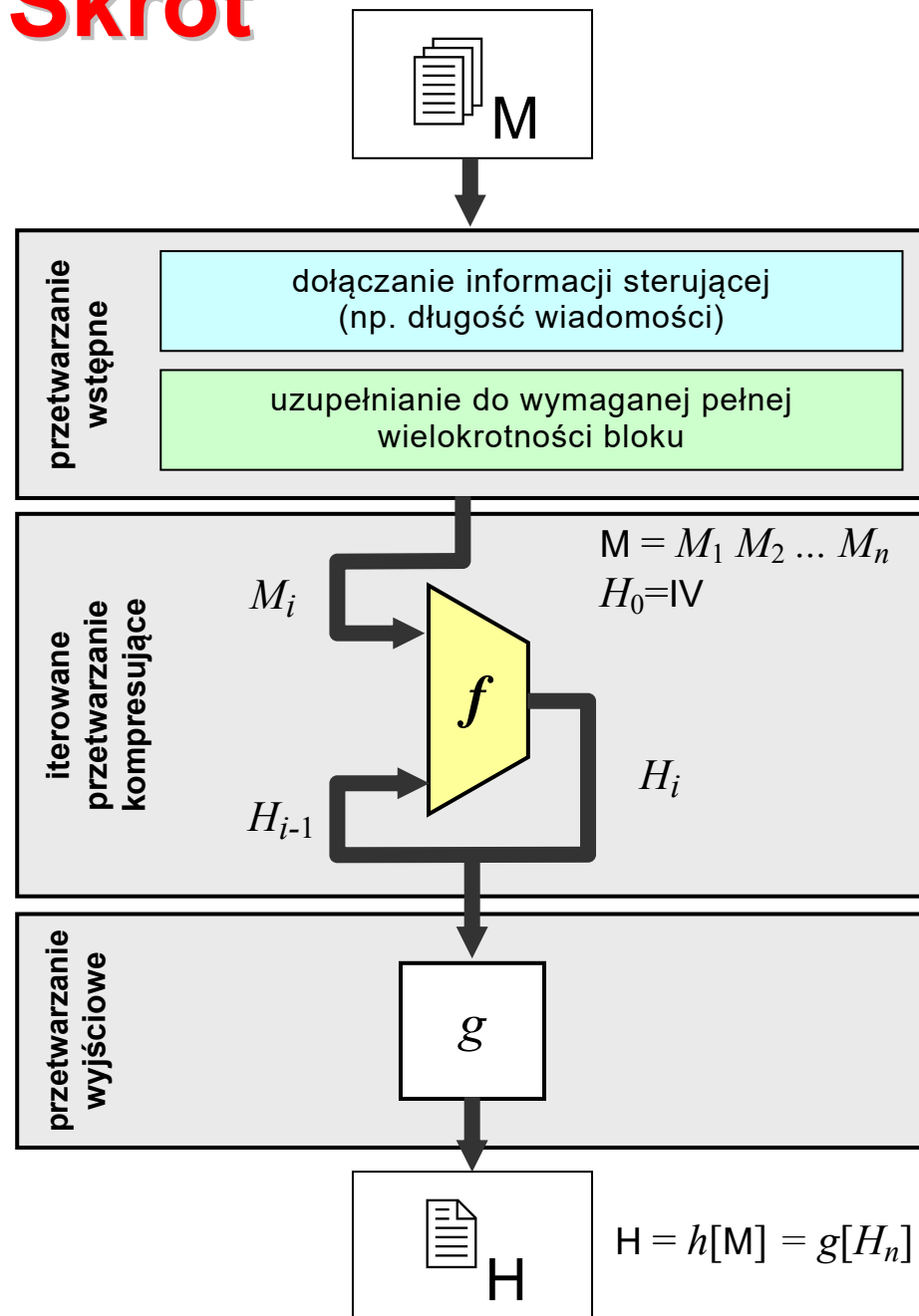
Skrót

Idea:



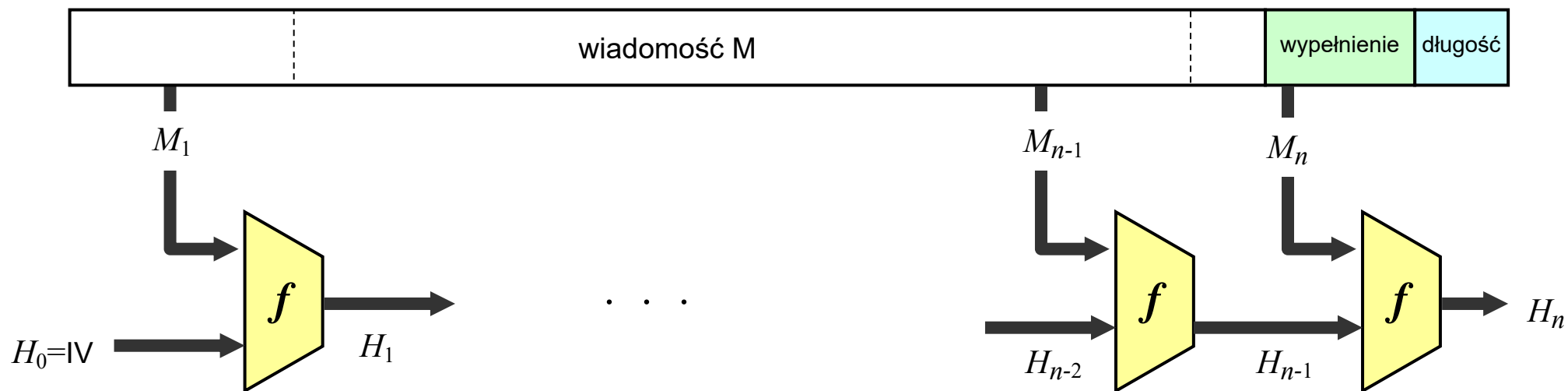
Skrót

Schemat:



Skrót

Merkle–Damgård:



➔ https://researchgate.net/publication/322094216_Merkle-Damgard_Construction_Method_and_Alternatives_A_Review

All in one



k_a
 K_A

M

H

$$E_{k_a}[H] = H'$$

$$E_{K_B}[M + H'] = S$$

M H'



S



$$D_{k_b}[S] = M + H'$$

M H'



H

=

H

$$D_{K_A}[H'] = H$$

poufność
integralność
autentyczność
niezaprzeczalność



k_b
 K_B

Algorytmy skrótu

MD (*Message Digest*)

- algorytmy MD i MD2 (Ron Rivest) powstały w latach '80.
- w 1990 Ralph Merkle zaproponował algorytm o nazwie Snefru – kilkukrotnie szybszy od MD2
- na to Rivest odpowiedział MD4 (RFC 1320) – wykorzystując operacje 32b
- w 1992 złamano Snefru i wykryto pewną słabość MD4 w wersji 2-rund
- Rivest wzmocnił algorytm otrzymując trochę wolniejszy MD5 (RFC1321)

MD5

- skrót 128b
- wektor IV 128b

Algorytmy skrótu

SHA (*Secure Hash Algorithm*)

- SHA został opracowany przez NSA (*National Security Agency*)
- podobny do MD4, ale skrót 160b

SHA-1

- szybko wykryto słabości w SHA-0 (ich natury nigdy nie opublikowano)
- i opracowano SHA-1, nieco wolniejszy od MD5
- ratyfikowany przez NIST jako element standardu DSS
- odporność na kolizje 160b skrótu – 2^{80} jest współcześnie uznawana za zbyt słabą

Algorytmy skrótu

SHA-2

- funkcje SHA-224 / SHA-256 / SHA-384 / SHA-512
- przystosowane do współpracy z kluczami 112b, 128b, 192b i 256b
- dają skróty 224b, 256b, 384b i 512b
- dość złożone obliczeniowo
- SHA-224 ma identyczny koszt co SHA-256
SHA-384 ma identyczny koszt co SHA-512

Algorytmy skrótu

SHA-3

- otwarty konkurs – zgłoszono ok. 60 algorytmów

http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo

- m.in. MD6 (Rivest) i Skein (→Treefish, Schneier),

- w październiku 2012 r. wyłoniono zwycięzcę – algorytm Keccak

<http://keccak.noekeon.org>

- nie stosuje schematu kompresji Merkle-Damgård → funkcja gąbki

- FIPS 202 (2015)

https://csrc.nist.gov/CSRC/media/Publications/fips/202/final/documents/fips_202_draft.pdf

Algorytmy skrótu

Wydajność:



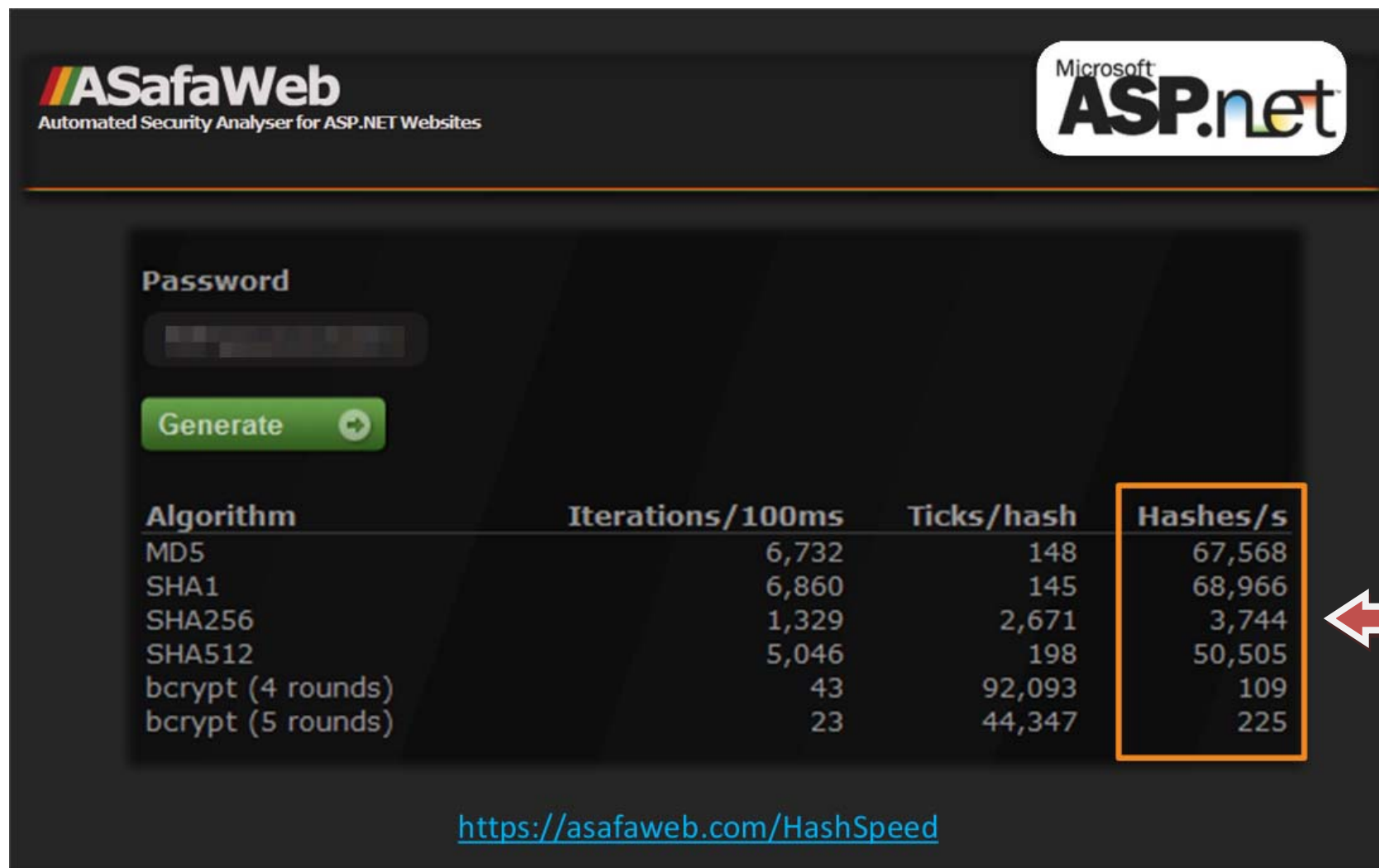
Algorytm	Iteracji	Czas
md5	10 000 000	5.924 sec
sha1	10 000 000	5.964 sec
sha2-256	10 000 000	6.056 sec
sha2-512	10 000 000	6.192 sec
bcrypt-5	1 000	3.112 sec
bcrypt-10	1 000	91.290 sec
bcrypt-11	1 000	182.410 sec

Python, hash-benchamrk.py
md5, sha – hashlib, bcrypt – passlib

Dual Core P8600 2.4Ghz, 4GB RAM
Ubuntu Linux 12.04 **64bit**

Algorytmy skrótu

Wydajność:



ASaFaWeb
Automated Security Analyser for ASP.NET Websites

Microsoft
ASP.NET

Password

Generate

Algorithm	Iterations/100ms	Ticks/hash	Hashes/s
MD5	6,732	148	67,568
SHA1	6,860	145	68,966
SHA256	1,329	2,671	3,744
SHA512	5,046	198	50,505
bcrypt (4 rounds)	43	92,093	109
bcrypt (5 rounds)	23	44,347	225

<https://asafaweb.com/HashSpeed>

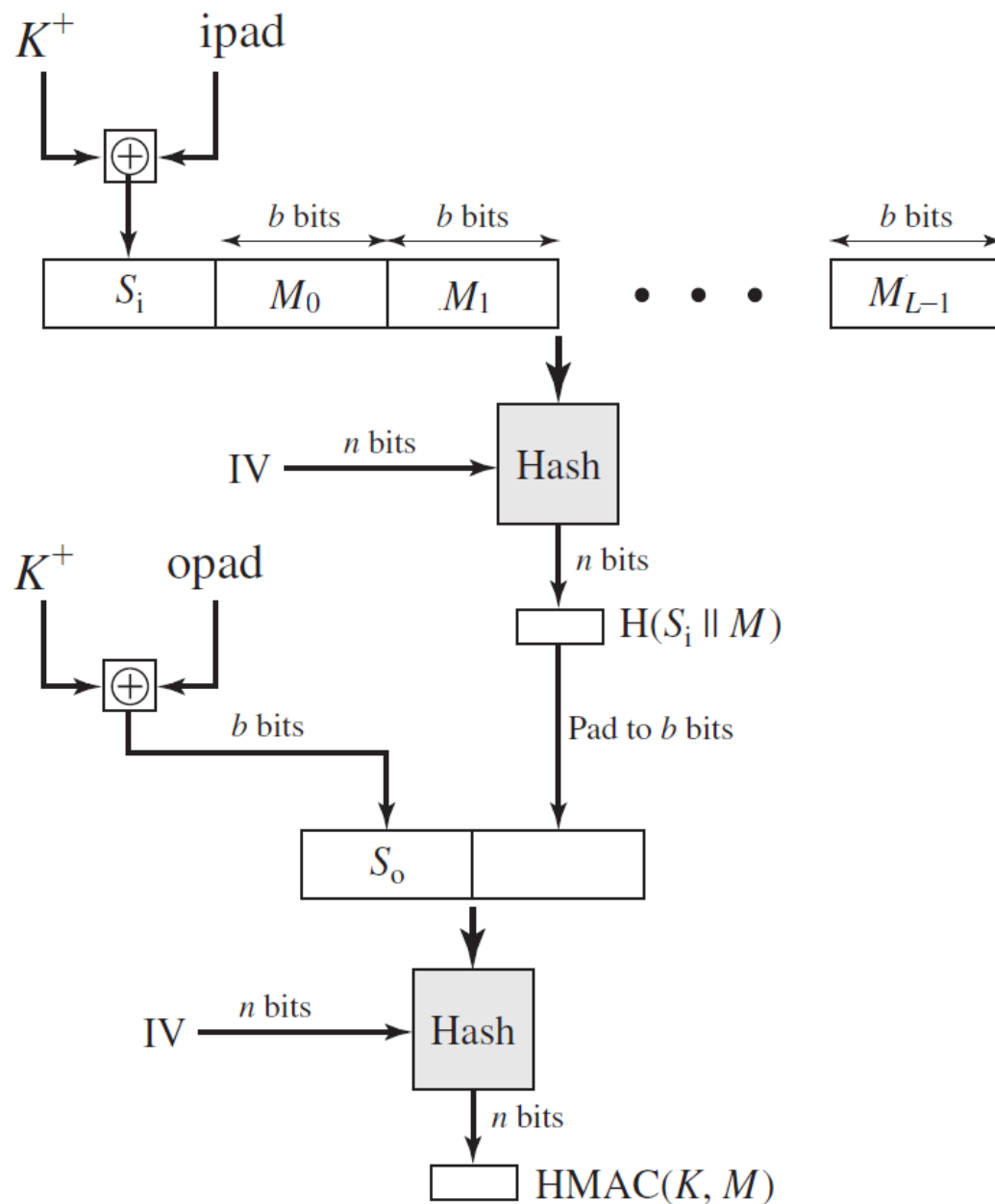
Autentyczność

HMAC (*Hashing MAC*) RFC 2104

- $\text{HMAC}_k[M]$
- standard FIPS PUB 198
- skrót MD5 (HMAC-MD5) lub SHA-1 (HMAC-SHA1)
- w praktyce dowolność: HMAC-SHA-256, ...
- uzyskiwany („szyfrowany”) z użyciem klucza symetrycznego k
- oferuje integralność i autentyczność (a niezaprzeczalność?)
- bardzo rozpowszechniony, np. IPsec i TLS

Autentyczność

HMAC



Autentyczność

Poly1305 authenticator, RFC 7539

- nowoczesny $\text{HMAC}_k[M]$
- opracowany przez Daniela J. Bernsteina,
twórcę nowych szyfrów Salsa20 i ChaCha20
- szybki
- i zyskujący na popularności

Autentyczność

ElGamal

- algorytm ten początkowo opracowano właśnie dla podpisu cyfrowego
- podpisem wiadomości M jest para $(a, b) ::$

$$a = g^k \bmod p$$

$$b :: M = (xa + kb) \bmod p-1 \quad (\text{rozszerzone równanie Euklidesa})$$

- weryfikacja podpisu cyfrowego:

$$\text{jeśli } (y^a a^b) \bmod p == g^M \bmod p$$

Autentyczność

DSA (*Digital Signature Algorithm*)

- pierwotnie odmiana algorytmu ElGamala opracowana przez NSA
- standard FIPS 186 (186-4 od 2003 r.)
- dostępny darmowo, także w zastosowaniach komercyjnych
- aktualnie – odmiany dla krzywych eliptycznych:
 - ECDSA
 - EdDSA (RFC 8032)
 - Ed25519 (krzywa Edwards25519)

Autentyczność

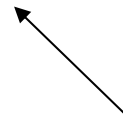
Wykorzystanie AES tylko do MAC

- AES-CMAC (CBC MAC)
- AES-GMAC

Wykorzystanie AES jednocześnie do szyfrowania i MAC

(Authenticated Encryption)

- AES-CCM (CM + CBC MAC)
- AES-GCM (GCM + GMAC)
- AES-Poly1305
- ChaCha20-Poly1305



TLS, SSH, IPsec, 802.1ae, 802.11i, 802.11ad

Authenticated Encryption

“Nil cryptographiae sine veritate”

Encrypted Cookies Are Not Enough

- Developers often encrypt cookie payloads assuming it cannot be changed
- Encryption **does not provide integrity!** Attackers can modify an encrypted cookie without knowing the key

```
def encryptCookie(payload, key, iv):  
    obj = AES.new(key, AES.MODE_CTR, iv)  
    str1 = padding(payload)  
    ciphertext = obj.encrypt(str1)  
    return ciphertext  
  
AuthCookieVal = encryptCookie("Role:Reviewer", "aiBuacoM8", "mee0epJee")
```

Bit-flip to Victory

Cookie payload = "Role:Reviewer" provides the cookie value (hex) below
set-cookie: auth=de6dd89e66232da8a4dac92845; ← This isn't signed!

Attacker:

By gathering cookies from various roles, looking for patterns and bit-flipping with XOR, a new valid cookie can be crafted without knowing the encryption key

de6dd89e66232da8a4dac92845 XOR 13011b000b

Cookie: auth=de6dd89e66302cb3a4d1 ← Outcome used to set attacker's cookie

Decrypts to "Role:Admin"

Zagrożenia

Ataki na funkcje skrótu

Kolizje w MD5:

2 ciągi po 128 B różniące się 6-cioma bajtami:

00000000	d1	31	dd	02	c5	e6	ee	c4	69	3d	9a	06	98	af	f9	5c
00000010	2f	ca	b5	87	12	46	7e	ab	40	04	58	3e	b8	fb	7f	89
00000020	55	ad	34	06	09	f4	b3	02	83	e4	88	83	25	71	41	5a
00000030	08	51	25	e8	f7	cd	c9	9f	d9	1d	bd	f2	80	37	3c	5b
00000040	96	0b	1d	d1	dc	41	7b	9c	e4	d8	97	f4	5a	65	55	d5
00000050	35	73	9a	c7	f0	eb	fd	0c	30	29	f1	66	d1	09	b1	8f
00000060	75	27	7f	79	30	d5	5c	eb	22	e8	ad	ba	79	cc	15	5c
00000070	ed	74	cb	dd	5f	c5	d3	6d	b1	9b	0a	d8	35	cc	a7	e3

MD5 = a4c0d35c95a63a805915367dcfe6b751

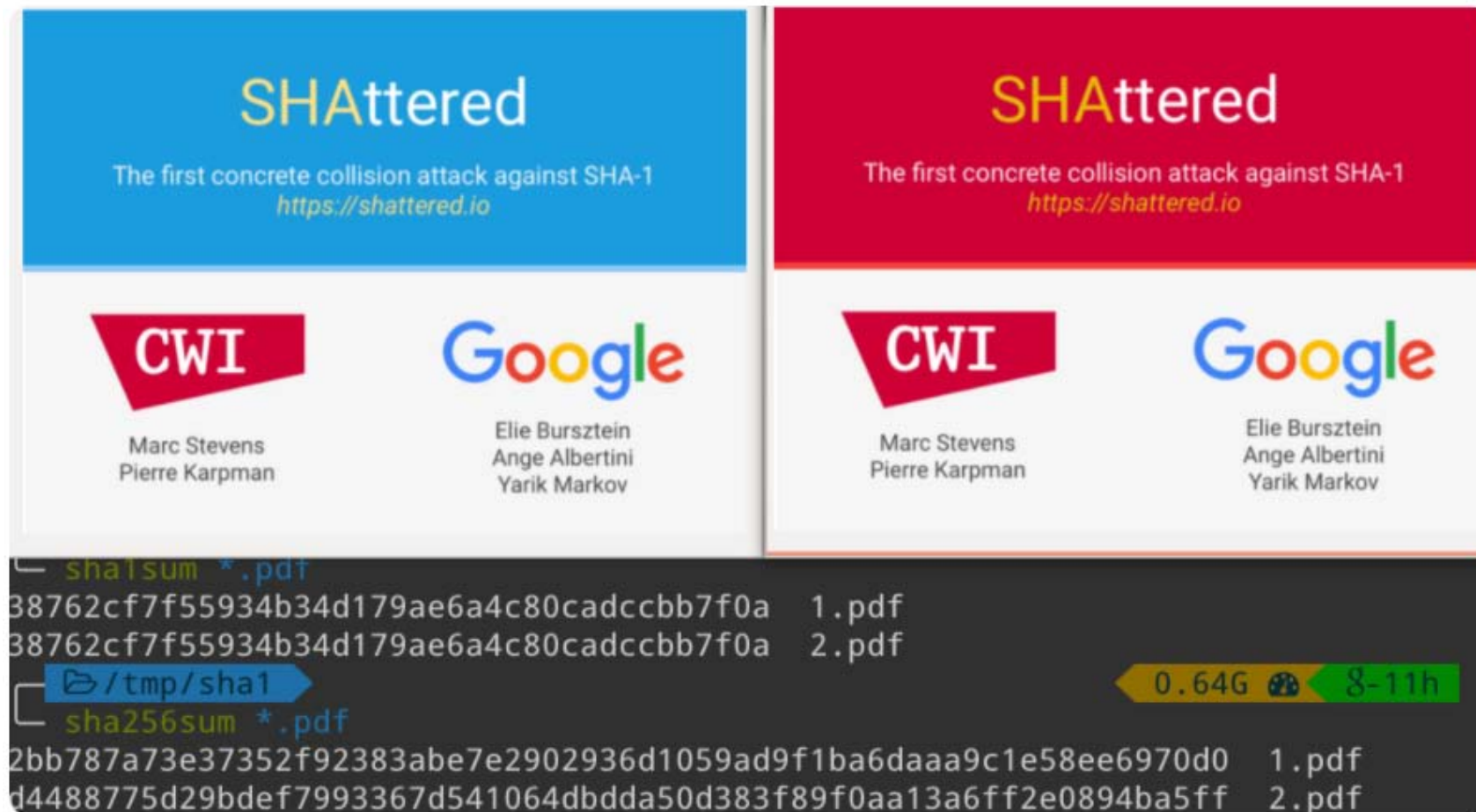
00000000	d1	31	dd	02	c5	e6	ee	c4	69	3d	9a	06	98	af	f9	5c
00000010	2f	ca	b5	07	12	46	7e	ab	40	04	58	3e	b8	fb	7f	89
00000020	55	ad	34	06	09	f4	b3	02	83	e4	88	83	25	f1	41	5a
00000030	08	51	25	e8	f7	cd	c9	9f	d9	1d	bd	72	80	37	3c	5b
00000040	96	0b	1d	d1	dc	41	7b	9c	e4	d8	97	f4	5a	65	55	d5
00000050	35	73	9a	47	f0	eb	fd	0c	30	29	f1	66	d1	09	b1	8f
00000060	75	27	7f	79	30	d5	5c	eb	22	e8	ad	ba	79	4c	15	5c
00000070	ed	74	cb	dd	5f	c5	d3	6d	b1	9b	0a	58	35	cc	a7	e3

MD5 = a4c0d35c95a63a805915367dcfe6b751

<http://www.mathstat.dal.ca/~selinger/md5collision/>

Ataki na funkcje skrótu

Kolizje w SHA-1:



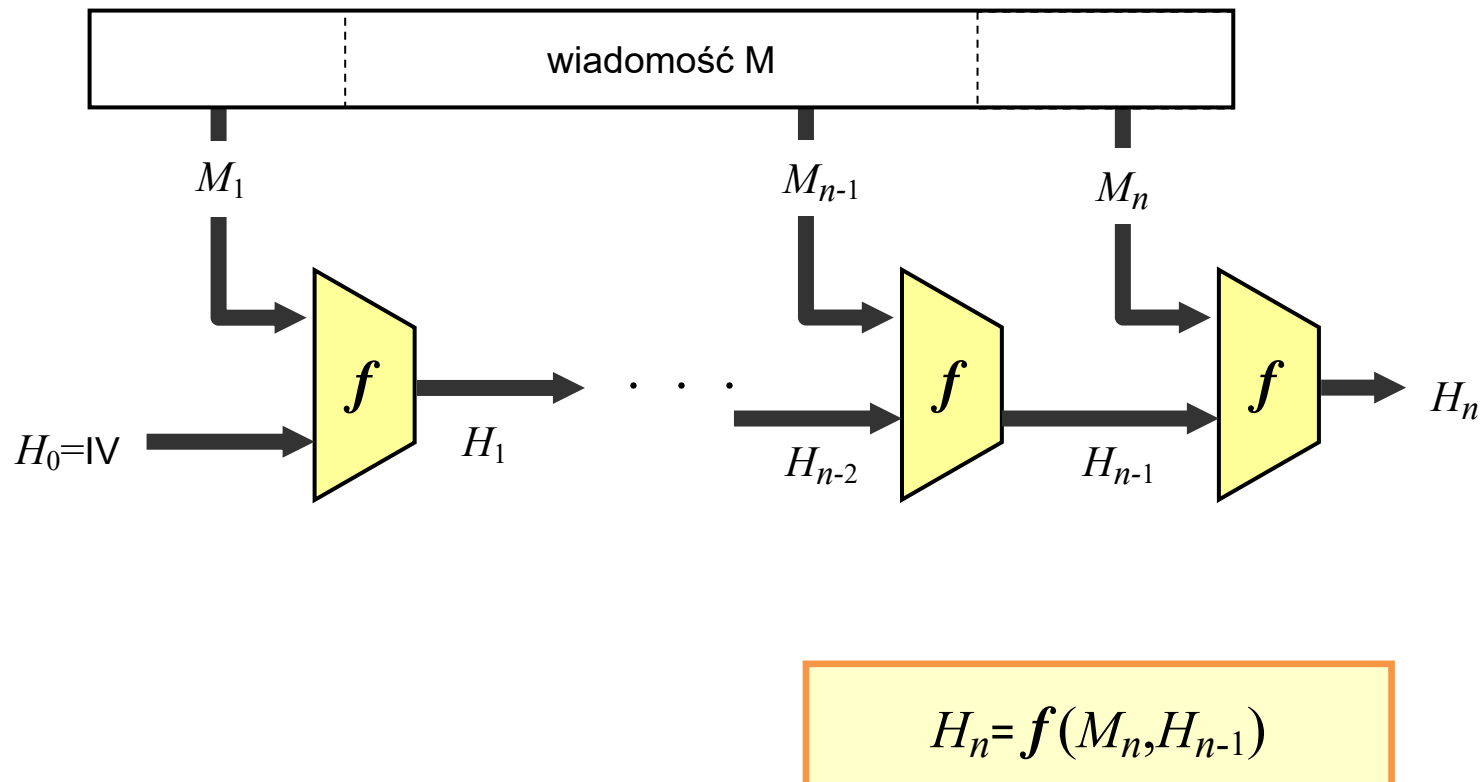
Ataki na funkcje skrótu

Inne kolizje:

- <https://eprint.iacr.org/2004/199.pdf>
- dla zastosowań MAC kolizje nie przesądzają jeszcze o całkowitej nieprzydatności funkcji skrótu

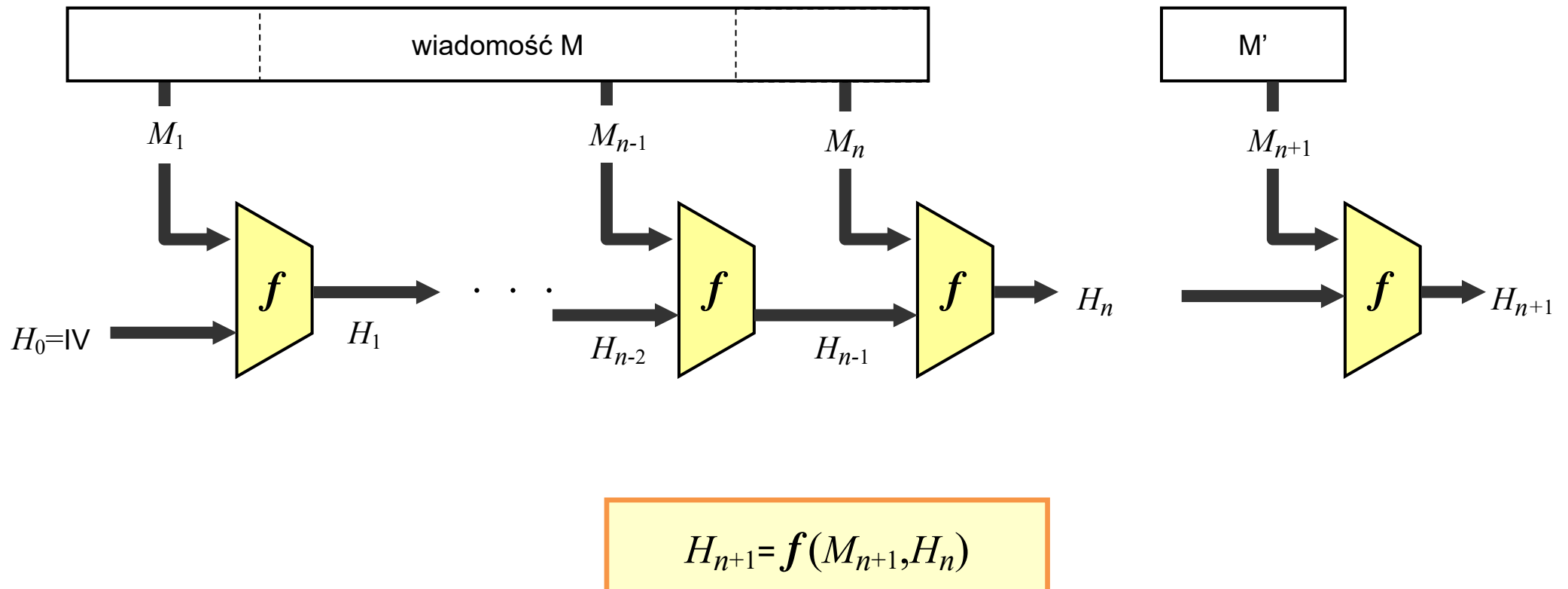
Ataki na funkcje skrót

Typowy skrót:



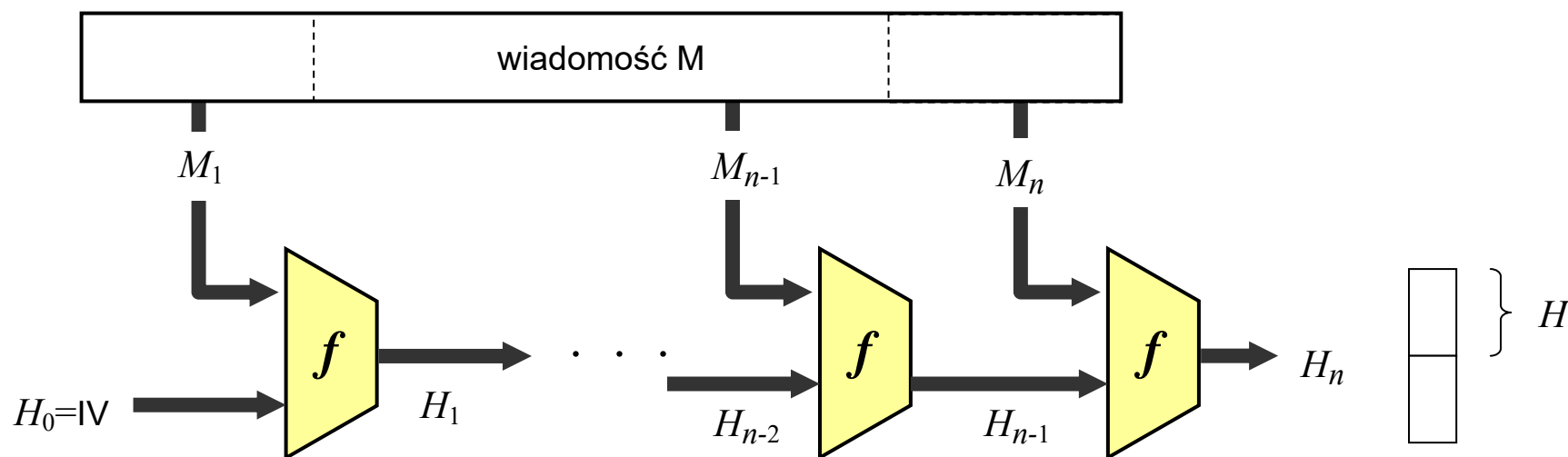
Ataki na funkcje skrótu

Length extension attack:



Ataki na funkcje skrótu

Obrona (→ SHA-3):

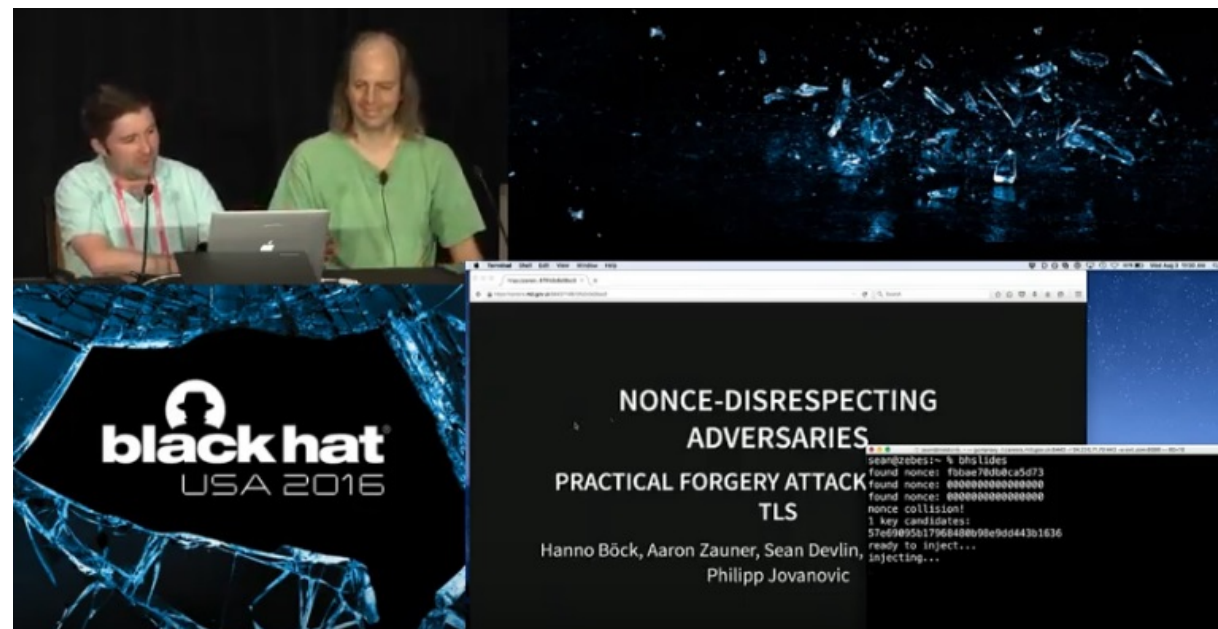


$$H_{n+1} = f(M_{n+1}, H_n = ?)$$

Problemy z losowością

IV = nonce !

- unikalność nonce jest kluczowa dla wielu algorytmów kryptograficznych (np. w trybie AES-CM przesądza o bezpieczeństwie)
 - niekiedy ważny jest też jednorodny rozkład (*uniform distribution*) wartości nonce (np. dla ECDSA, choć już nie dla EdDSA)
- "Nonce-Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS"



Problemy z losowością

Przykładowy RNG

```
RNGCryptoServiceProvider random_generator;  
  
byte[] nonce = new Byte[32];    // 32B * 8b = 256-bit nonce  
  
random_generator = new RNGCryptoServiceProvider();  
  
random_generator.GetNonZeroBytes ( nonce );
```

Problemy z losowością

Zjawiskowy RNG

```
int getRandomNumber()  
{  
    return 4; // chosen by fair dice roll.  
              // guaranteed to be random.  
}
```

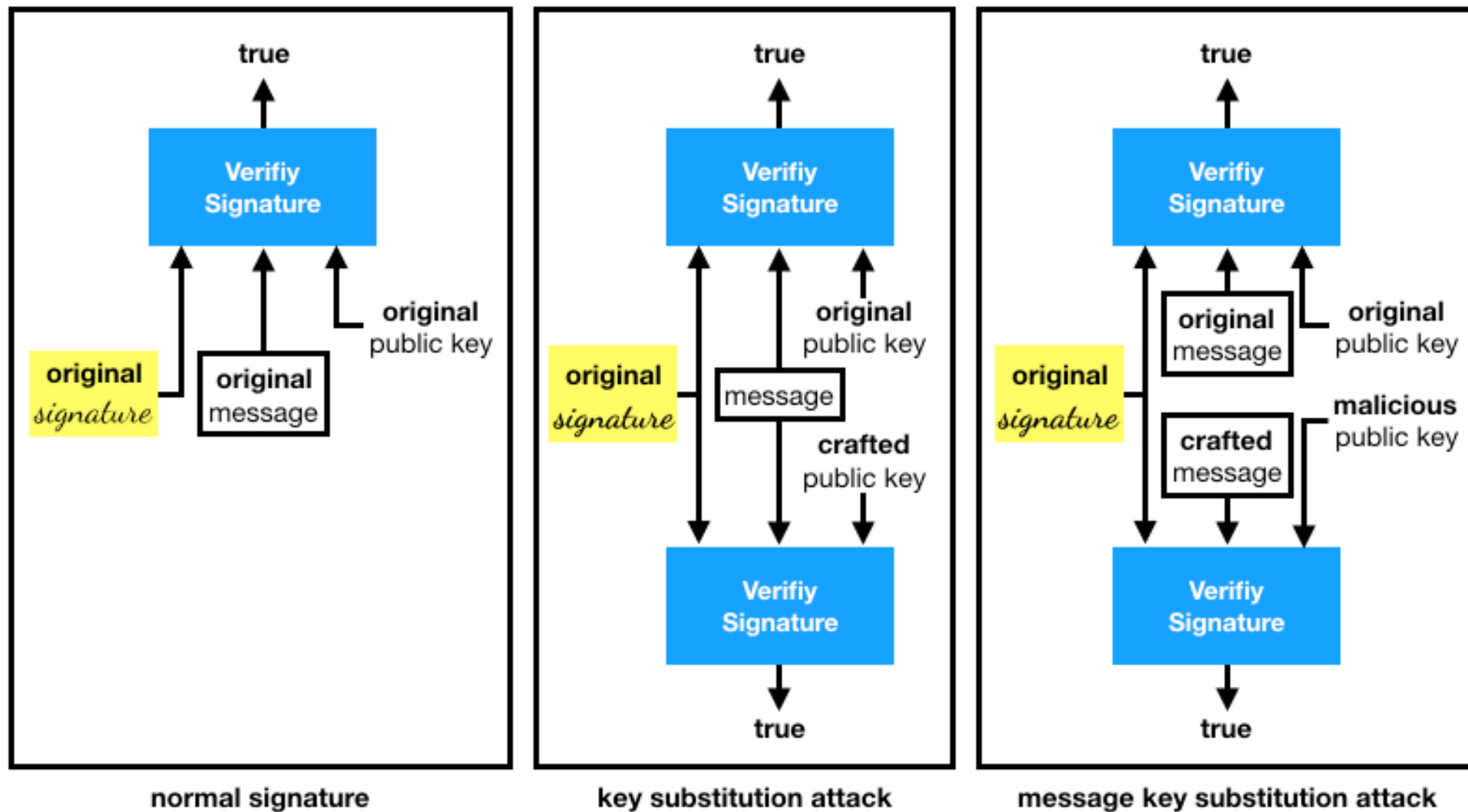
Source: <http://www.xkcd.com>

Problemy z losowością

- nowe wymagania: nonce misuse resistant algorithms (→ Lightweight Crypto for IoT)
- Synthetic IV (SIV):

$$(\text{pre-})\text{nonce} \oplus \text{plaintext } M_1 = \text{IV}$$

Problemy z kluczami



Zarządzanie kluczami

Problemy

- tajny klucz symetryczny?
- gdzie go bezpiecznie przechowywać?
- jak go przekazać partnerowi komunikacji?
- jak go regularnie zmieniać w sposób bezpieczny?

Dystrybucja klucza

Metoda Diffiego-Hellmana

Whitfield Diffie, Martin Hellman

Idea:

- ustalamy wspólny jednorazowy **klucz sesji** (dla każdej sesji inny)
- na potrzeby ustalenia klucza sesji wykorzystamy model kryptografii asymetrycznej
- na scenę wkraczają ponownie liczby pierwsze
- DH wykorzystuje multiplikatywną grupę modulo p – \mathbb{Z}_p^*
- α jest elementem pierwotnym grupy \mathbb{Z}_p^* (α generujące całą grupę)
- czyli zamiast $1, \dots, p-1$ możemy grupę traktować jako $1, \alpha, \alpha^2, \dots, \alpha^{p-2}$
- istnieje także odmiana ECDH dla krzywych eliptycznych

Metoda Diffiego-Hellmana



k_a

$$K_A = \alpha^{k_a} \bmod p$$

Wspólny klucz sesji

α, p



k_b

$$K_B = \alpha^{k_b} \bmod p$$

$$\Phi_A = K_B^{k_a} = (\alpha^{k_b})^{k_a} \bmod p$$

$$\Phi_B = K_A^{k_b} = (\alpha^{k_a})^{k_b} \bmod p$$

$$\Phi_A = \alpha^{(k_a \cdot k_b)} \bmod p = \Phi_B$$

Φ

klucz sesji

Φ

Metoda Diffiego-Hellmana

Atak *Man-in-the-Middle*:

- Edziu, znając α , może skutecznie wkroczyć na etapie negocjacji klucza
- Alicja i Bolek będą porozumiewać się poprzez Edzia za pomocą kluczy, które – jak im się będzie wydawać – wymienili ze sobą

Przypadek szczególny:

- co jeśli Edziu przechwytyjąc komunikację zastąpi $K_A = \alpha^{k_a}$ oraz $K_B = \alpha^{k_b}$ przez wartość 1?

Obrona:

- DH + authentication = MAC kluczy K_A , K_B i Φ

Klucze sesji

Uogólnienie metody dystrybucji klucza dla wielu stron:

Group Key Exchange (GKE)

- wielu uczestników sesji
- dodatkowe podatności i zagrożenia
(fault attacks, side channel attacks, cold boot attacks)
- nowa sesja przy każdej zmianie składu grupy

Stateful Group Key Exchange (stGKE)

- w negocjacji klucza sesji wykorzystuje utajniony składnik stanu poprzedniej sesji

Klucze sesji

Ataki aktywne:

- trwają badania nad protokołami KE odpornymi na ataki aktywne:
 - Authenticated Key Exchange (AKE)
 - Authenticated Group Key Exchange (AGKE)
 - Stateful Authenticated Group Key Exchange (stAGKE)

Dystrybucja kluczy publicznych

Dystrybucja kluczy publicznych

Warianty

1. pobranie klucza bezpośrednio od właściciela B
 2. pobranie klucza z centralnej bazy danych
 3. pobranie z własnej prywatnej bazy danych zapamiętanego wcześniej klucza pozyskanego sposobem 1 lub 2
- w ogólności istnieje ryzyko podstawienia przez nieuczciwą osobę E własnego klucza pod klucz użytkownika B

Certyfikaty kluczy publicznych

Certyfikacja

- w celu uniknięcia podstawienia klucza publicznego stosuje się certyfikację
- certyfikat jest podpisany przez osobę (instytucję) godną zaufania
- nazywaną urzędem certyfikującym CA (*Certification Authority*)
- urząd certyfikujący CA potwierdza, iż informacja opisująca użytkownika B jest prawdziwa a klucz publiczny faktycznie do niego należy
- certyfikat zawiera podstawowe dane identyfikujące właściciela
- posiada też okres ważności
- niezależnie od okresu ważności klucze mogą zostać uznane za niepoprawne – urząd poświadczający CA musi przechowywać listę niepoprawnych i nieaktualnych certyfikatów

Certyfikaty kluczy publicznych

Struktura podstawowa typowego certyfikatu

Numer seryjny	wartość niepowtarzalna identyfikująca certyfikat
Wystawca	urząd certyfikujący CA, który wystawił certyfikat
Okres ważności	początkowa i końcowa data ważności certyfikatu
Podmiot	nazwa podmiotu, dla którego stworzono certyfikat
Klucz publiczny podmiotu (algorytm, parametry, klucz)	klucz publiczny podmiotu z identyfikatorem algorytmu
Algorytm podpisu (algorytm, parametry)	algorytm stosowany do podpisania certyfikatu
Podpis	podpis cyfrowy urzędu certyfikującego CA

Certyfikaty kluczy publicznych

Przykład certyfikatu

Nazwa podmiotu	_____
Nazwa pospolita	ekonto.put.poznan.pl
Jednostka organizacyjna	Network Management Centre
Organizacja	Poznan University of Technology
Państwo	PL
Nazwa wystawcy	_____
Państwo	PL
Organizacja	Poznan University of Technology
Jednostka organizacyjna	Network Management Centre
Nazwa pospolita	PUT Root Certification Authority
Ważność	_____
Nieważny przed	7.07.2008, 08:17:14 (czas środkowoeuropejski letni)
Nieważny po	17.07.2010, 08:17:14 (czas środkowoeuropejski letni)
Informacje o kluczu publicznym	_____
	D4:73:74:29:6B:A4:65:32:E4:1D:0A:8F:B1:47:A5:72:80:F5:85:7D:AF:FF:04:31:7C:39:A8:1A:34:C7:D5:D3:6F:A1:1...
Algorytm	RSA
Rozmiar klucza	1024
Wykładnik	65537
Różne	_____
Numer seryjny	12:BB:17:08:14
Algorytm podpisu	SHA-1 with RSA Encryption

Certyfikaty kluczy publicznych

Standard ITU-T X.509

Budowa certyfikatu X.509 v.3:

- struktura podstawowa uzupełniona o:
 - sposób wykorzystania klucza (przeznaczenie, np. do szyfrowania danych, do szyfrowania kluczy, do uzgadniania kluczy, do podpisywania danych, do osiągnięcia niezaprzeczalności)
 - informacje o polityce certyfikacji wydawcy certyfikatu (np. ograniczenia długości ścieżki certyfikacji, punkty dystrybucji list certyfikatów unieważnionych)
- oraz opcjonalna możliwość tworzenia dodatkowych pól / parametrów identyfikacyjnych, wg potrzeb komunikujących się podmiotów
- typ MIME: application/x-x509-ca-cert

HOMEWORK

=

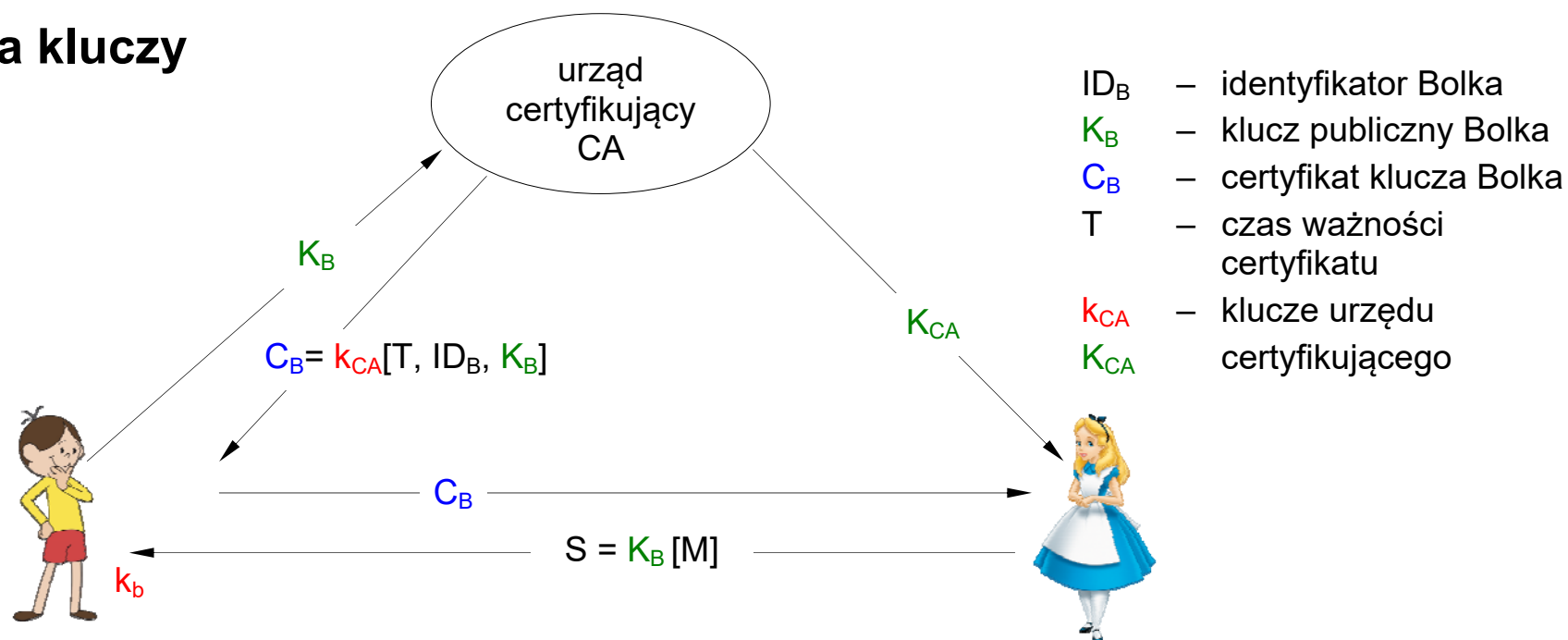
Half Of My Energy Wasted On Random Knowledge

→ **Standard IEEE 1609.2**



Certyfikaty kluczy publicznych

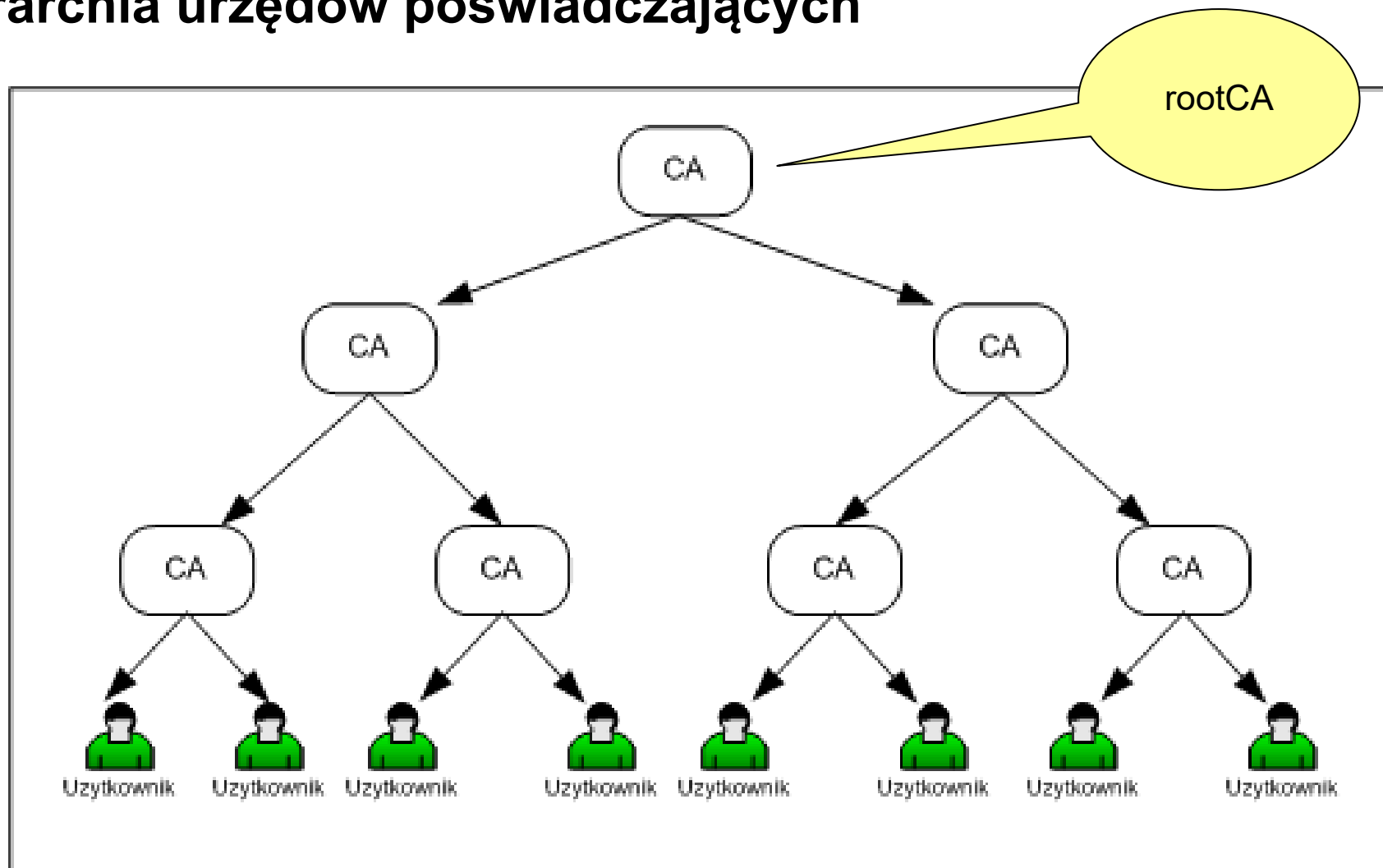
Dystrybucja kluczy



- w celu uzyskania certyfikatu użytkownik zwraca się do urzędu certyfikującego CA dostarczając mu swój klucz publiczny
- do zweryfikowania certyfikatu niezbędny jest klucz publiczny CA

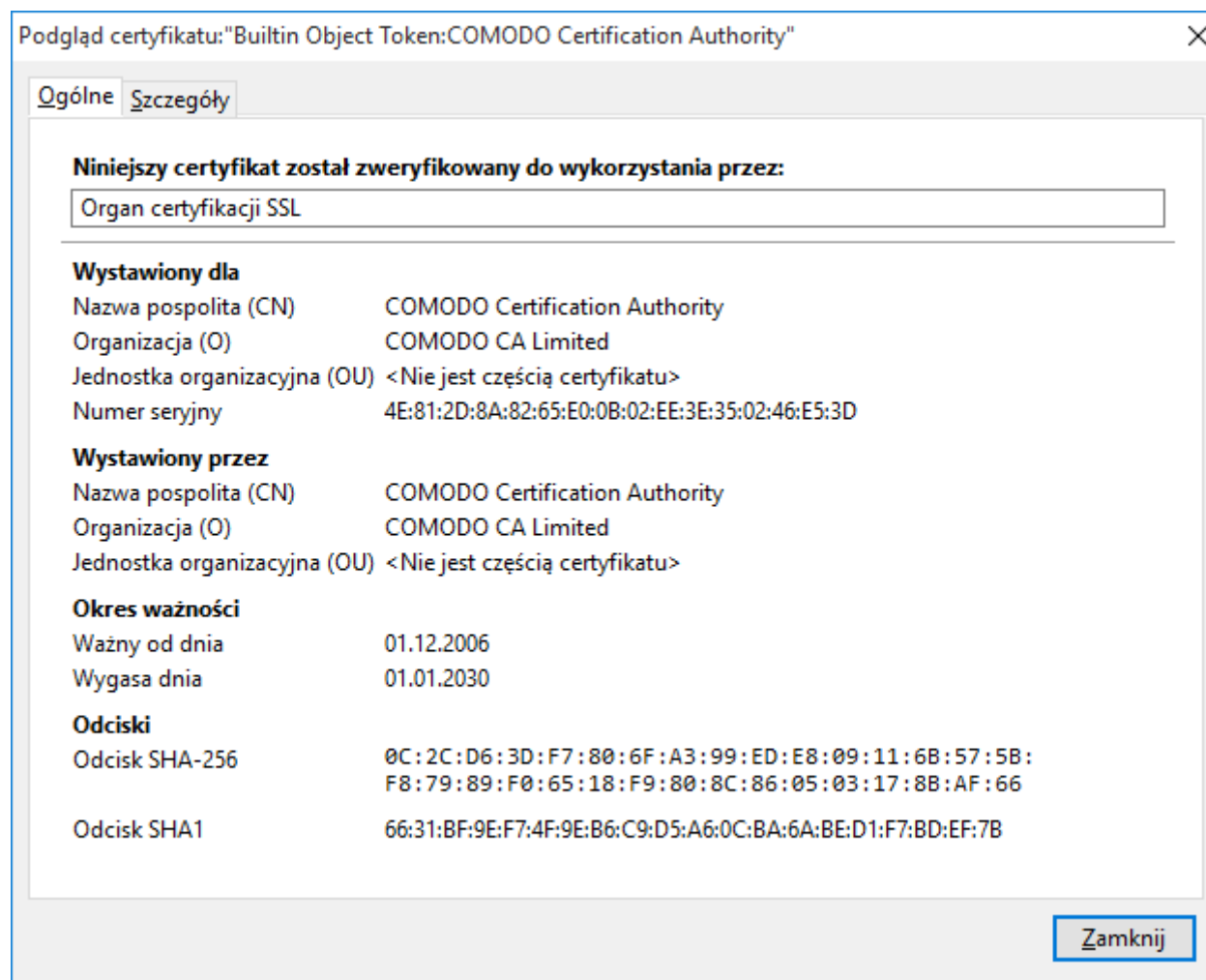
Certyfikaty kluczy publicznych

Hierarchia urzędów poświadczających



Certyfikaty kluczy publicznych

Hierarchia urzędów poświadczających



Infrastruktura kluczy publicznych

PKI (*Public Key Infrastructure*)

sprzęt, oprogramowanie, ludzie, polityki i procedury

konieczne do utworzenia, zarządzania, przechowywania, dystrybucji i unieważniania certyfikatów kluczy publicznych

- PKI oferuje często dodatkowe certyfikaty, np. Certyfikaty Znacznika Czasu (dla wiarygodnego potwierdzania czasu)

Infrastruktura kluczy publicznych

Komponenty PKI

- urzędy CA
- punkty rejestrujące (CAFE = CA Front End), poręczające zgodność kluczy z identyfikatorami (lub innymi atrybutami) posiadaczy certyfikatów
- repozytoria przechowujące i udostępniające certyfikaty i listy unieważnień (CABE = CA Back End)
- protokoły służące do wymiany informacji niezbędnych do właściwego zarządzania infrastrukturą kluczy publicznych: CMP (*Certificate Management Protocol*), SCEP (*Simple Certificate Enrollment Protocol*), SCVP (*Simple Certificate Validation Protocol*), OCSP (*Online Certificate Status Protocol*), ...

Infrastruktura kluczy publicznych

Standardy PKCS (*Public Key Cryptography Standard*)

- PKCS#7 – format podpisu cyfrowego (→ RFC 2315)
 - PKCS#10 – format żądań wystawienia certyfikatów
 - PKCS#12 – format składowania certyfikatów X.509 i kluczy prywatnych
-
- PKCS#1 – standard RSA
 - PKCS#2 – standard Diffiego-Hellmana
 - PKCS#5 – Password-based Encryption Standard (PBKDF2 → RFC 2898)
 - PKCS#11 – API do sprzętowych modułów kryptograficznych (HSM, TPM)

Infrastruktura kluczy publicznych

Polska

Podpis elektroniczny *kwalikowany*


- zostaje złożony przy użyciu certyfikatu kwalifikowanego
- i *bezpiecznego urządzenia* (SSCD – *Security Signature Creation Device*, norma CWA 14169 Europejskiego Komitetu Standaryzacji)
- wywołuje skutki prawne równoważne podpisowi własnoręcznemu

Podobne rozwiązania

- e-Dowód osobisty – podpis osobisty → edowod.gov.pl
- ePUAP – profil zaufany

Login 



Use eID of your country 

Choose country 

Wybierz sposób logowania

Bezpieczny i darmowy dostęp do usług publicznych

Profil Zaufany

Bezpłatne narzędzie, dzięki któremu załatwisz sprawy urzędowe online i podpiszesz dokumenty elektronicznie.

e-dowód

Dowód osobisty z warstwą elektroniczną. Użyj aplikacji mobilnej albo czytnika podłączonego do komputera.

mojID



inteligo



Infrastruktura kluczy publicznych

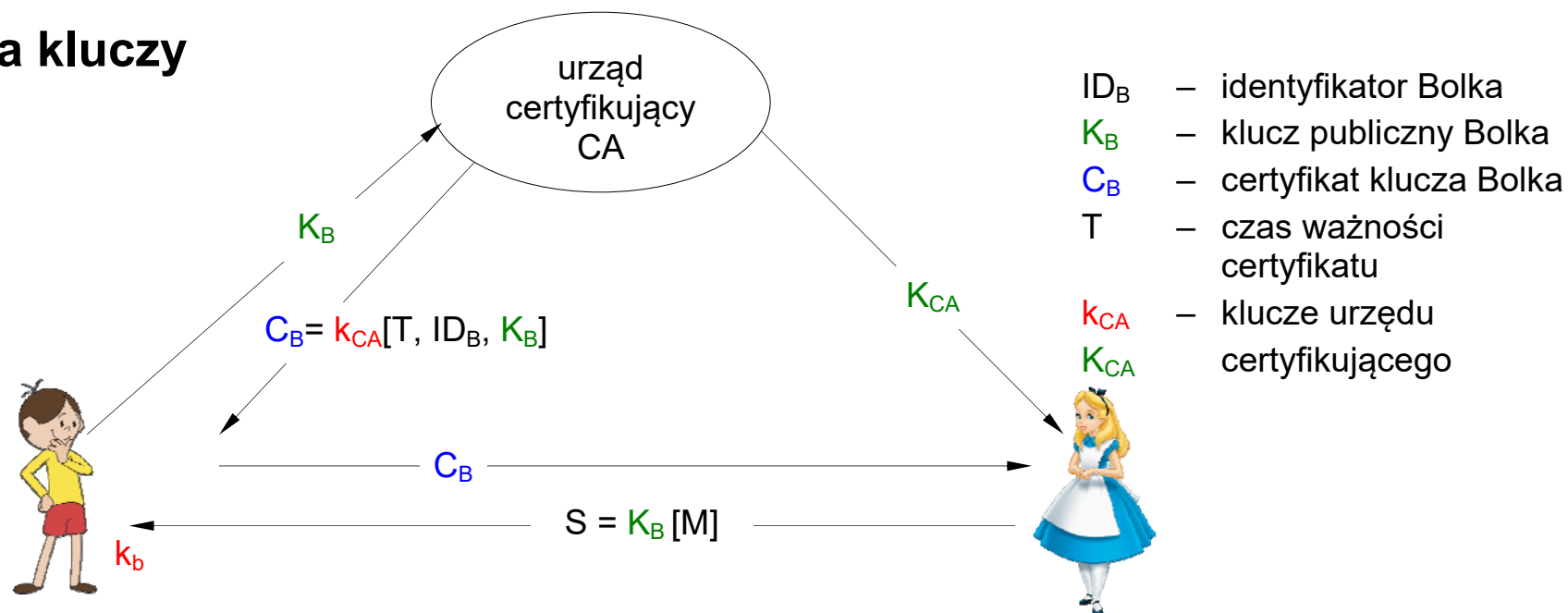
Ceny

	<i>Cena</i>	<i>Okres ważności</i>
Private Email	0 zł	(3 miesiące)
Certum Silver	50 zł	(rok)
Certum Silver - odnowienie	25 zł	(rok)
Certum Gold	200 zł	(rok)
Certum Gold - odnowienie	100 zł	(rok)
Certum Platinum (karta+czytnik)	700 zł	(2 lata)
Certum Platinum - odnowienie	200 zł	(2 lata)

	<i>Cena</i>	<i>Okres ważności</i>
Private Web Server	0 zł	(min. 3 miesiące)
Enterprise Web Server	500 zł	(rok)
Enterprise Web Server - odnowienie	250 zł	(rok)
Wildcard Domain	1000 zł	(rok)
Wildcard Domain - odnowienie	500 zł	(rok)
Trusted Web Server	1000 zł	(2 lata)
Trusted Web Server - odnowienie	500 zł	(2 lata)

Certyfikaty kluczy publicznych

Dystrybucja kluczy



- w celu uzyskania certyfikatu użytkownik zwraca się do urzędu certyfikującego CA dostarczając mu swój klucz publiczny
- do zweryfikowania certyfikatu niezbędny jest klucz publiczny CA
- użytkownik może przesłać swój certyfikat do innych użytkowników w celu poświadczenia jego autentyczności (Web of Trust)

Certyfikaty kluczy publicznych

Web of Trust



<https://xkcd.com>

Zastosowania kryptografii

Zastosowania kryptografii

Kryptograficzne zabezpieczenie danych

- ogromna ilość aplikacji szyfrowania plików
- ... i całego systemu plików:
 - moduł jądra Linux *cryptoloop* – szyfrowanie całego systemu plików na poziomie jądra za pomocą urządzenia blokowego `/dev/loop[1-8]`

```
mount -t ext4 crypto.raw /mnt/crypto -oencryption=aes-256
```

- EncFS – tworzy z wybranego katalogu wirtualny system plików w przestrzeni użytkownika, korzysta z modułu jądra FUSE (*Filesystem in USErspace*)

```
encfs ~/.crypto.vfs ~/tajne_dane
```

```
fusermount -u ~/tajne_dane
```

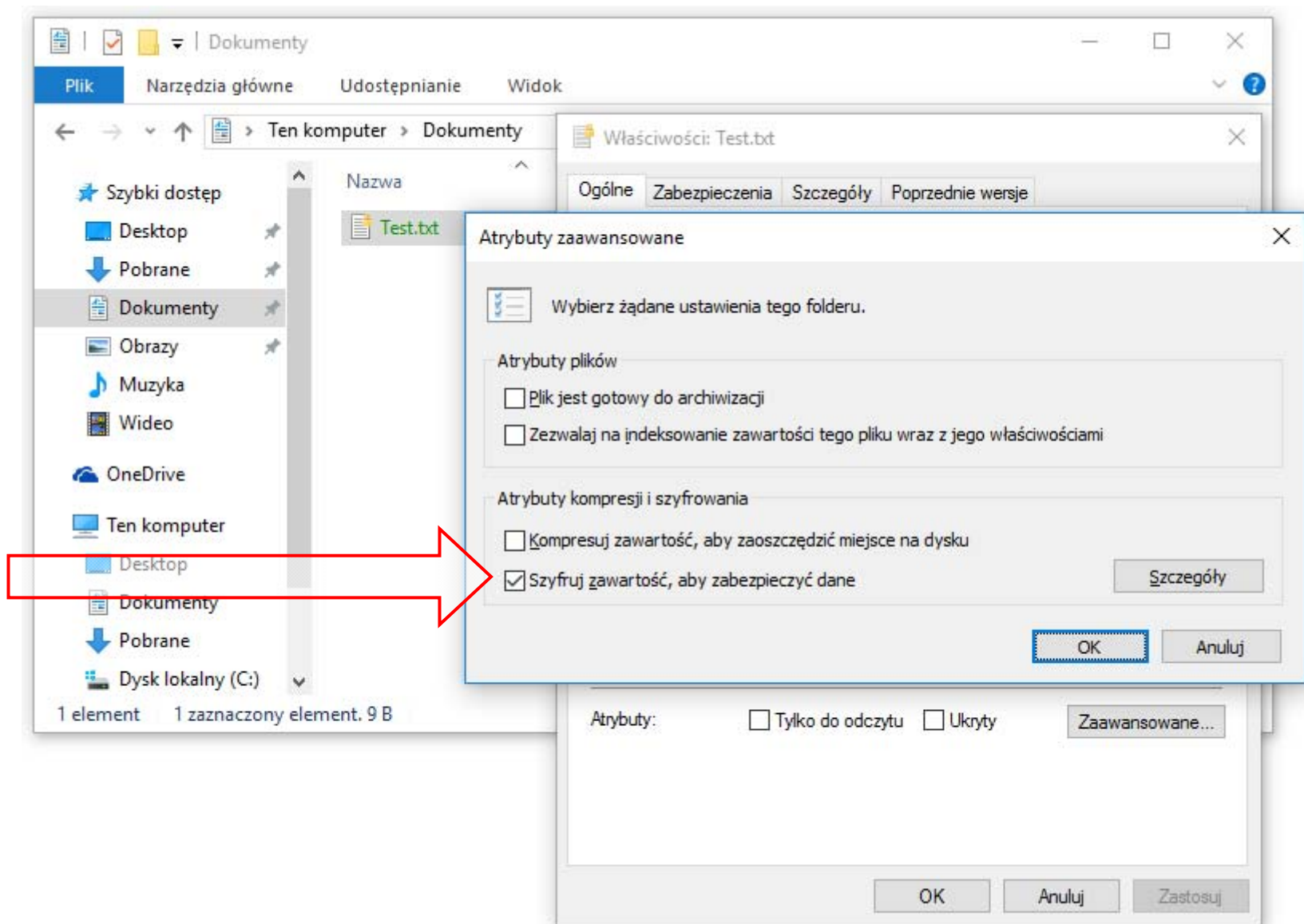
- dm-crypt i LUKS (*Linux Unified Key Setup*)

Zastosowania kryptografii

EFS

EFS (*Encrypted File System*) – usługa systemowa działa przezroczystie dla aplikacji

(DESX, 3DES,) AES

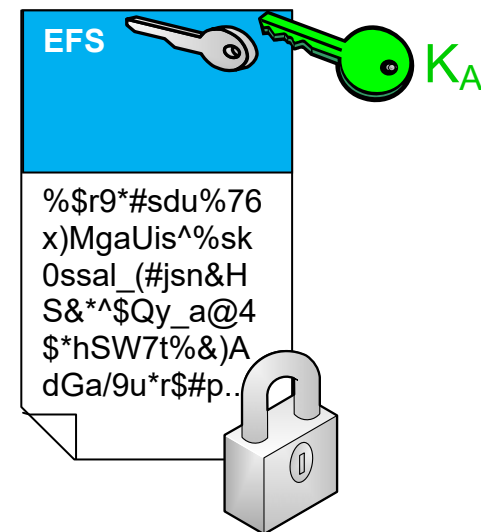


Zastosowania kryptografii

Nagłówek zaszyfrowanego pliku

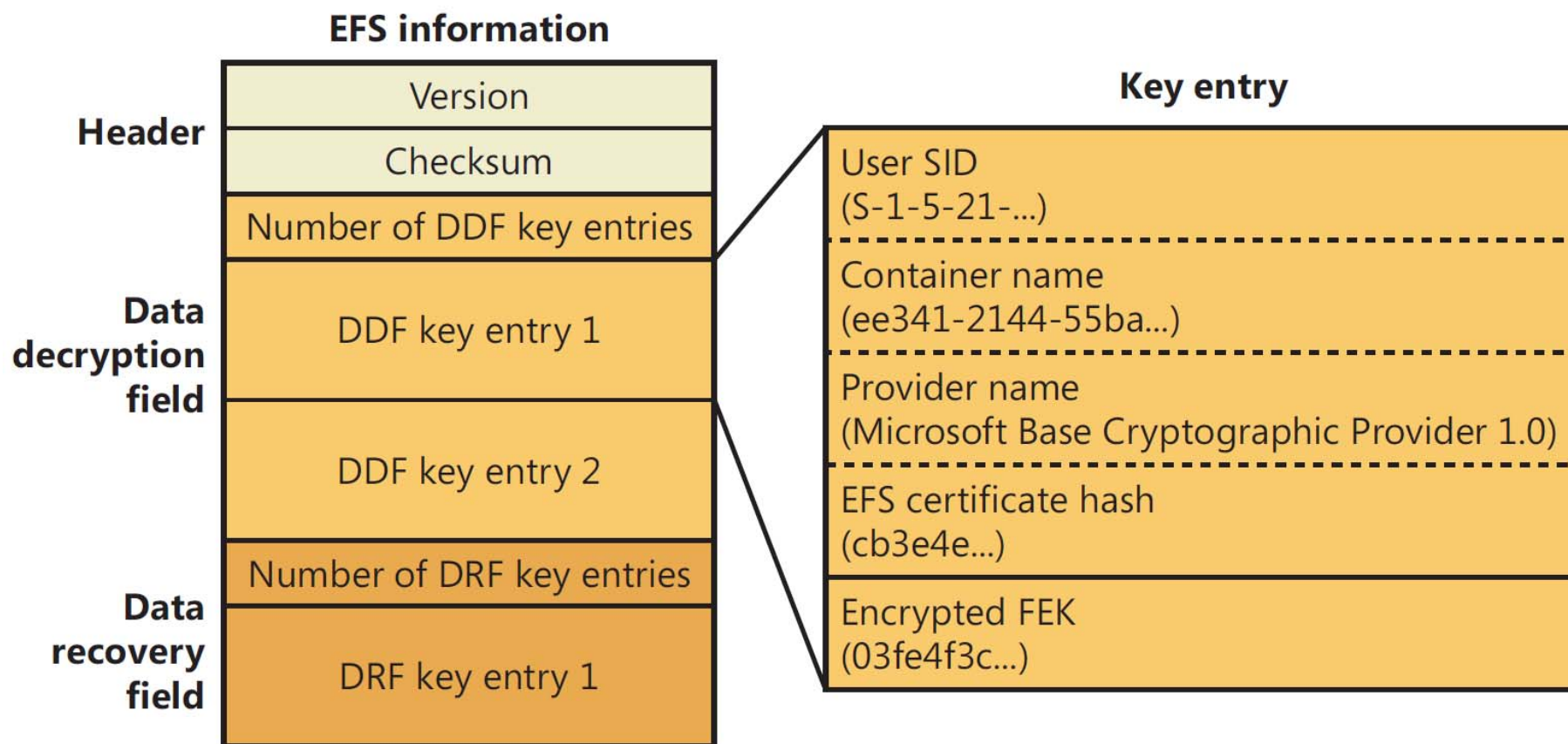
klucz symetryczny FEK zaszyfrowany kluczem prywatnym (RSA)

- właściciela pliku



Zastosowania kryptografii

Nagłówek zaszyfrowanego pliku

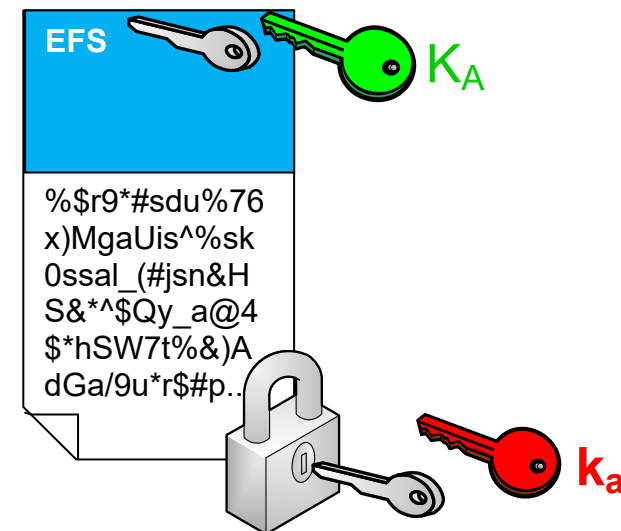


Zastosowania kryptografii

Nagłówek zaszyfrowanego pliku

klucz symetryczny FEK zaszyfrowany kluczem prywatnym (RSA)

- właściciela pliku

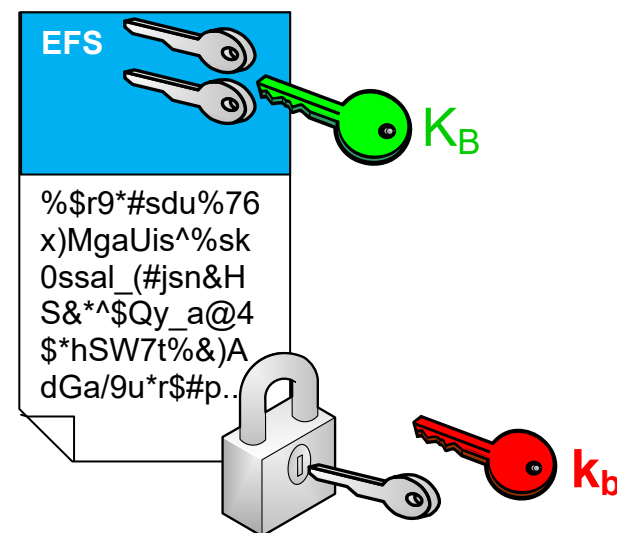


Zastosowania kryptografii

Nagłówek zaszyfrowanego pliku

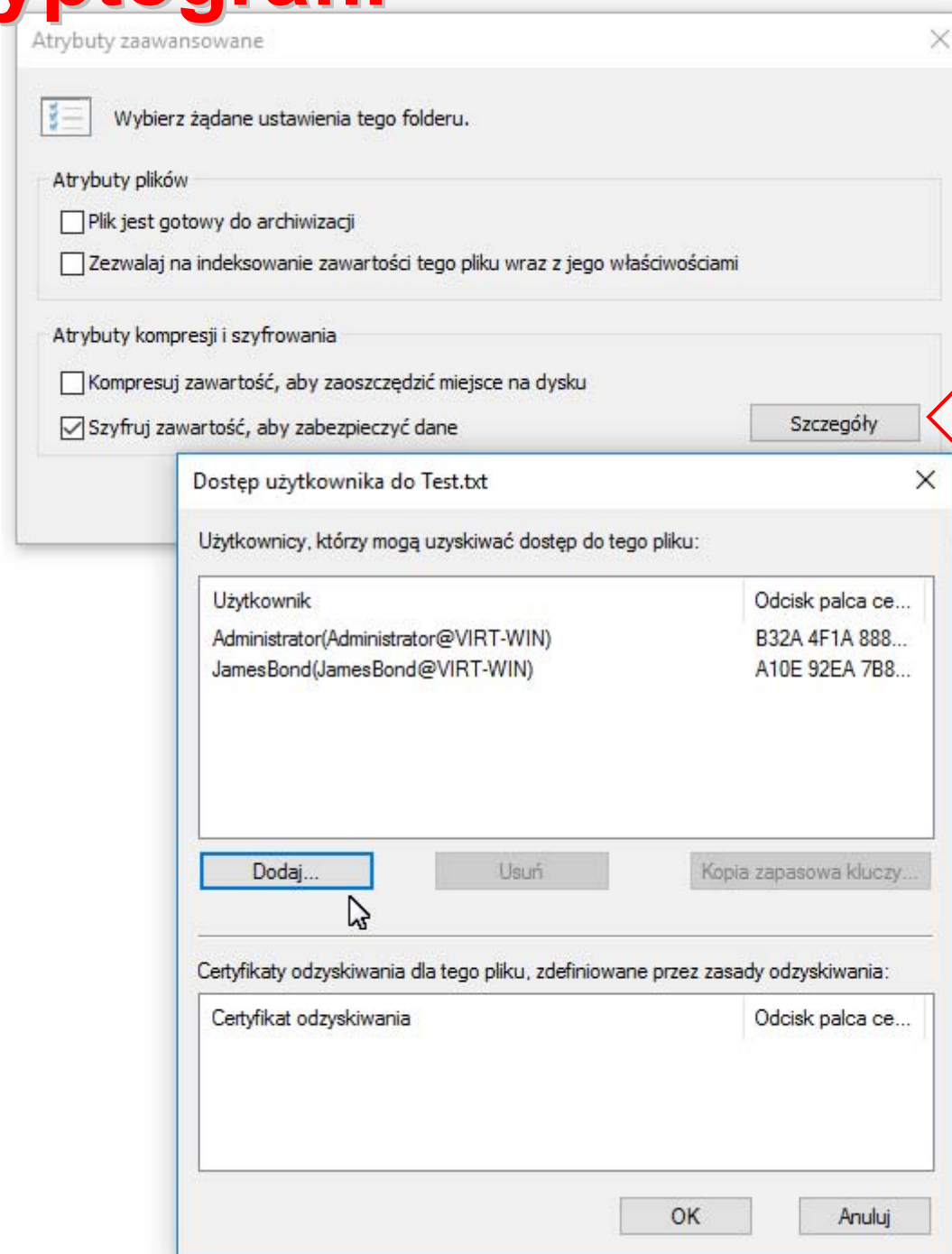
klucz symetryczny FEK zaszyfrowany kluczem prywatnym (RSA)

- właściciela pliku
- każdego współużytkownika



Zastosowania kryptografii

Współużytkowanie zaszyfrowanego pliku

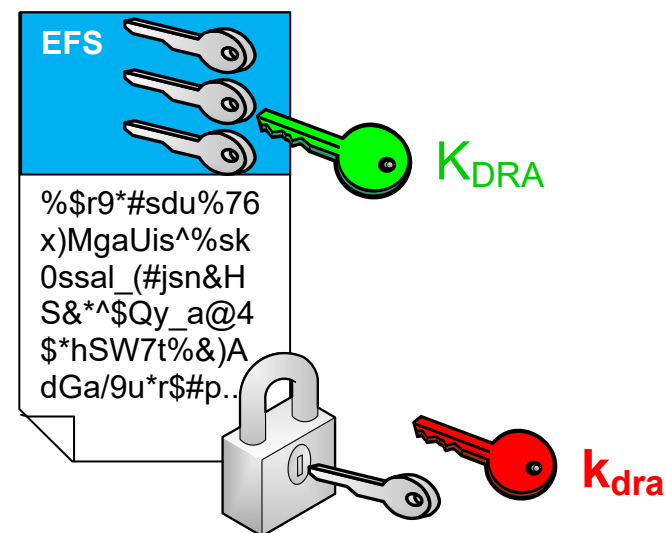


Zastosowania kryptografii

Nagłówek zaszyfrowanego pliku

klucz symetryczny FEK zaszyfrowany kluczem prywatnym (RSA)

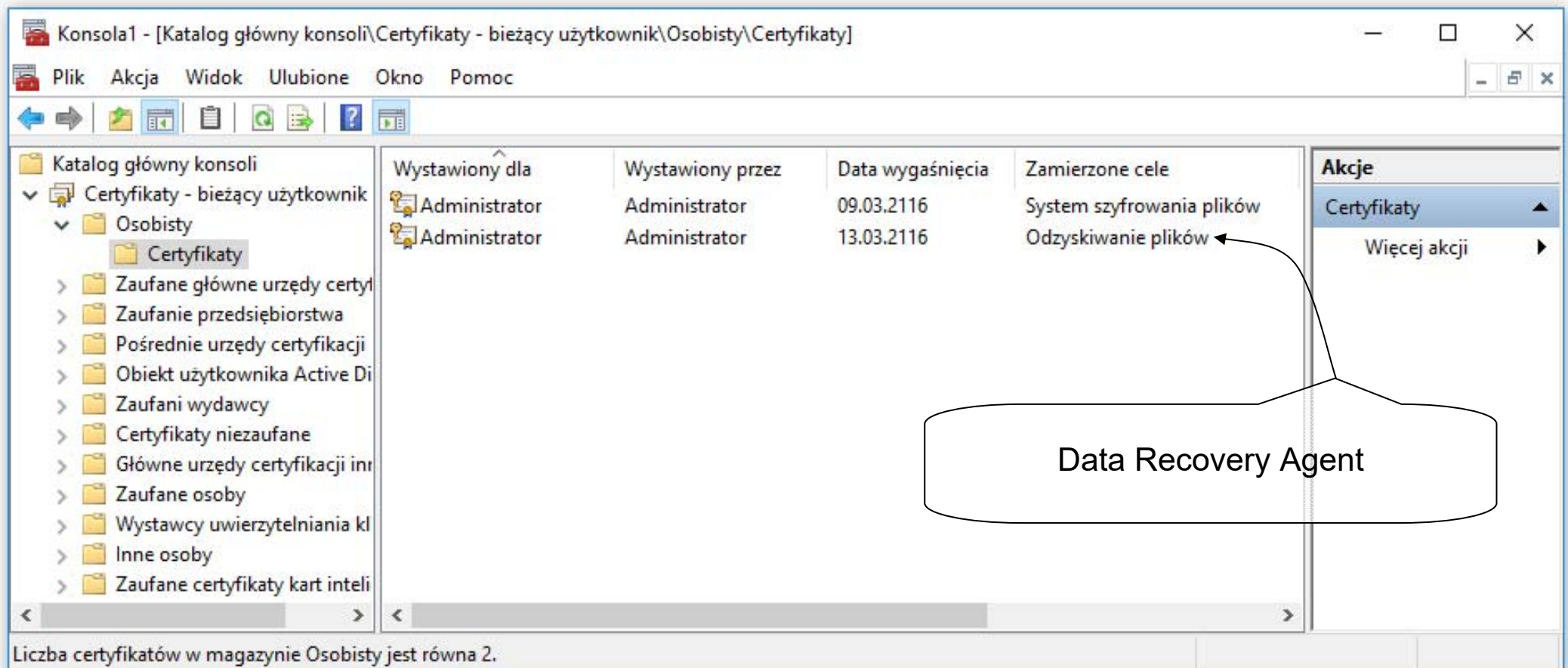
- właściciela pliku
- każdego współużytkownika
- każdego DRA
(Data Recovery Agent)



Zastosowania kryptografii

Klucze EFS

- klucze przechowywane są w postaci *self-signed certificates*



Zastosowania kryptografii



Problemy implementacyjne

Przykład: aplikacje szyfrujące dla Androida

- 9 przetestowanych aplikacji (“top developers”) ze sklepu Google Play
- żadna nie szyfruje całości pliku
- większość tak naprawdę nie szyfruje
- żadna nie stosuje Password-Based Key Derivation Function (PBKDF2)
- PIN jest tylko kamuflażem
- a co z oryginalnym plikiem?

“An Introduction to Security Challenges in User-Facing Cryptographic Software”, G. Paul, J. Irvine,
w *“Cybersecurity and Privacy – Bridging the Gap”*, River Publishers, 2017

Zastosowania kryptografii

Kryptograficzne zabezpieczenie komunikacji

Protokoły komunikacyjne:

- SSH (*Secure Shell*)
- SSL (*Secure Sockets Layer*) / TLS (*Transport Layer Security*) / DTLS (*Datagram TLS*)

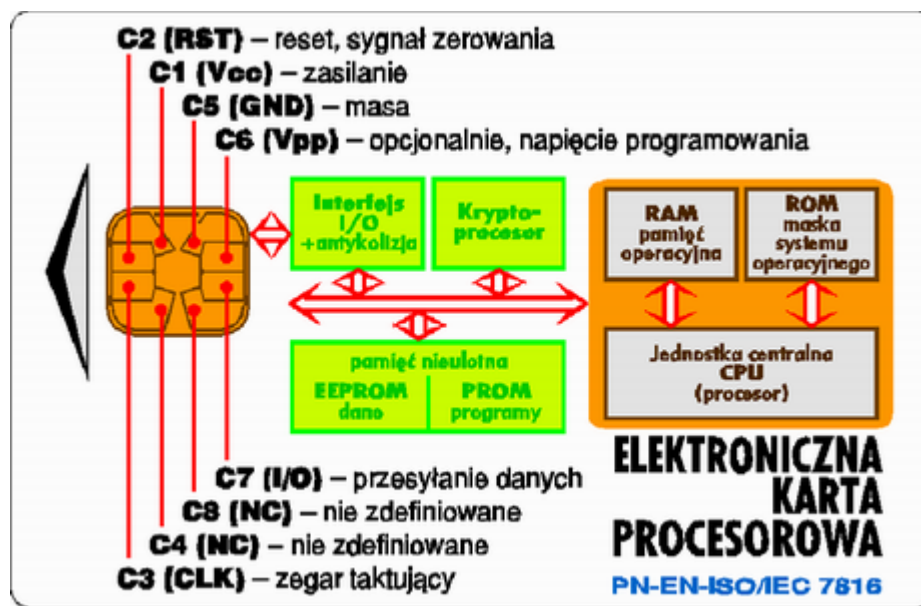
Komponenty aplikacji użytkowych:

- poczta elektroniczna: PGP, S/MIME, MIME/PGP, MSP
- komunikatory: OTR (Off-The-Record Messaging), Signal, KeyBase, ...
- WWW: HTTP/2

Zastosowania kryptografii

Tokeny kryptograficzne fizyczne i wirtualne

- kryptograficzne żetony (*cryptographic tokens*) uwierzytelniające tożsamość, autoryzujące dostęp i operacje, np. smart cards:



- kryptowaluta, np. Bitcoin, Litecoin, Ethereum, Monero, ...

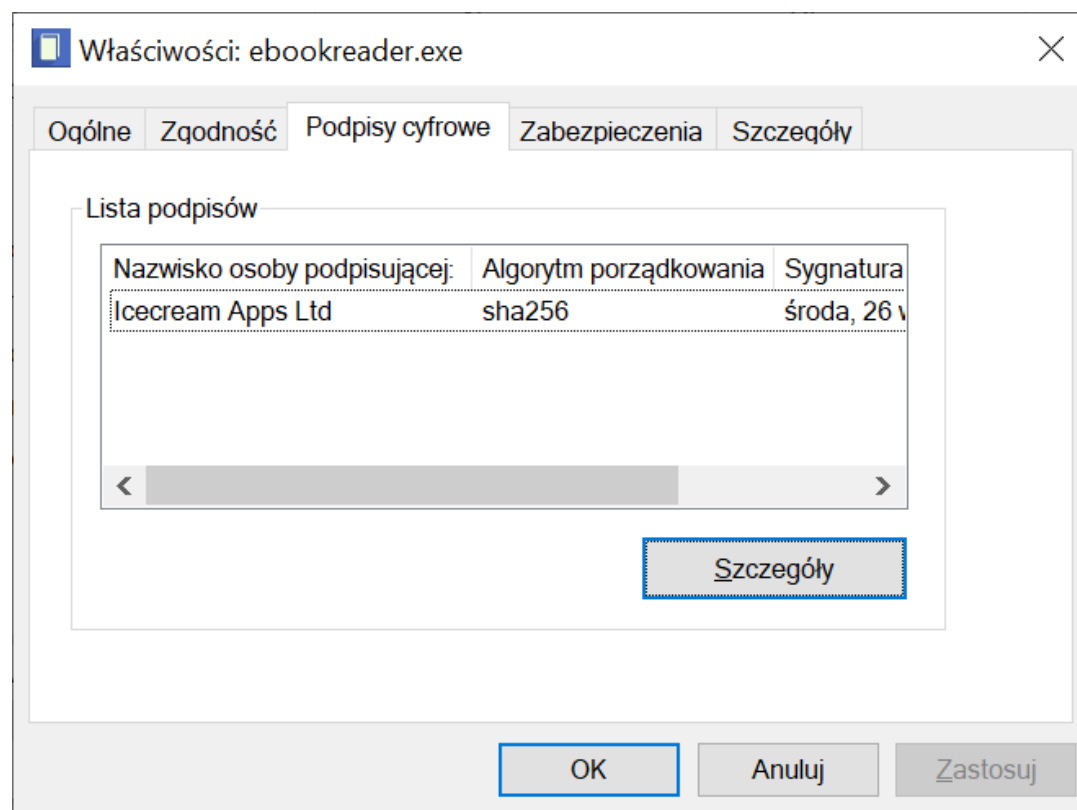
→ <https://blockchaindemo.io/>



Zastosowania kryptografii

Software Code Integrity (authenticode)

- user-mode code integrity (UMCI) = PE executables
- kernel-mode code integrity (KMCI) = drivers
- Windows 10 DeviceGuard



Zastosowania kryptografii

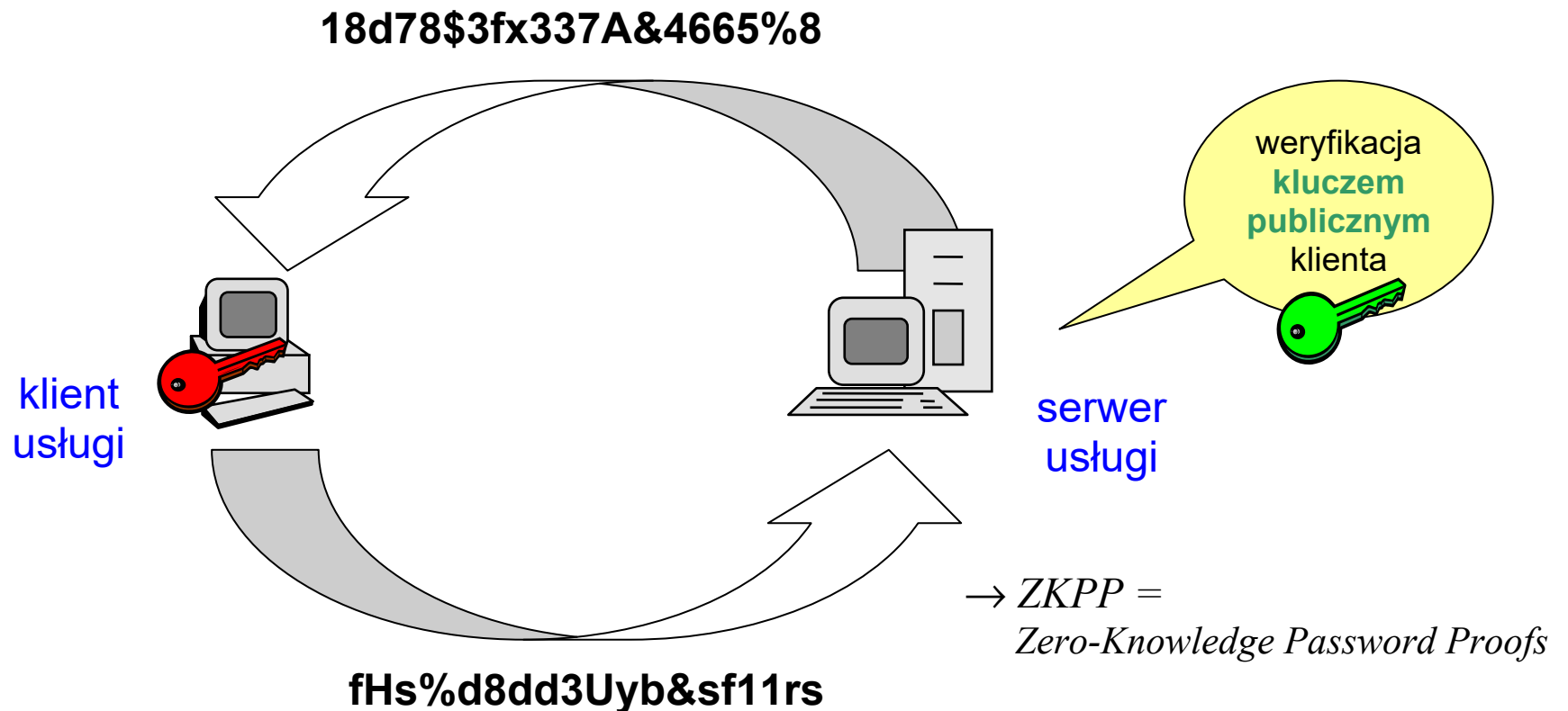
Secure Boot

- OS Secure Boot
 - UEFI Secure Boot
 - Platform Secure Boot (Verified Boot, Measured Boot)
 - digitally signed Secure Boot executables (PE)
 - UEFI certificate databases: db, dbx
- ➔ how are those databases protected from unauthorized modification?

Uwierzytelnianie

Metoda zwołanie-odzew (*challenge-response*):

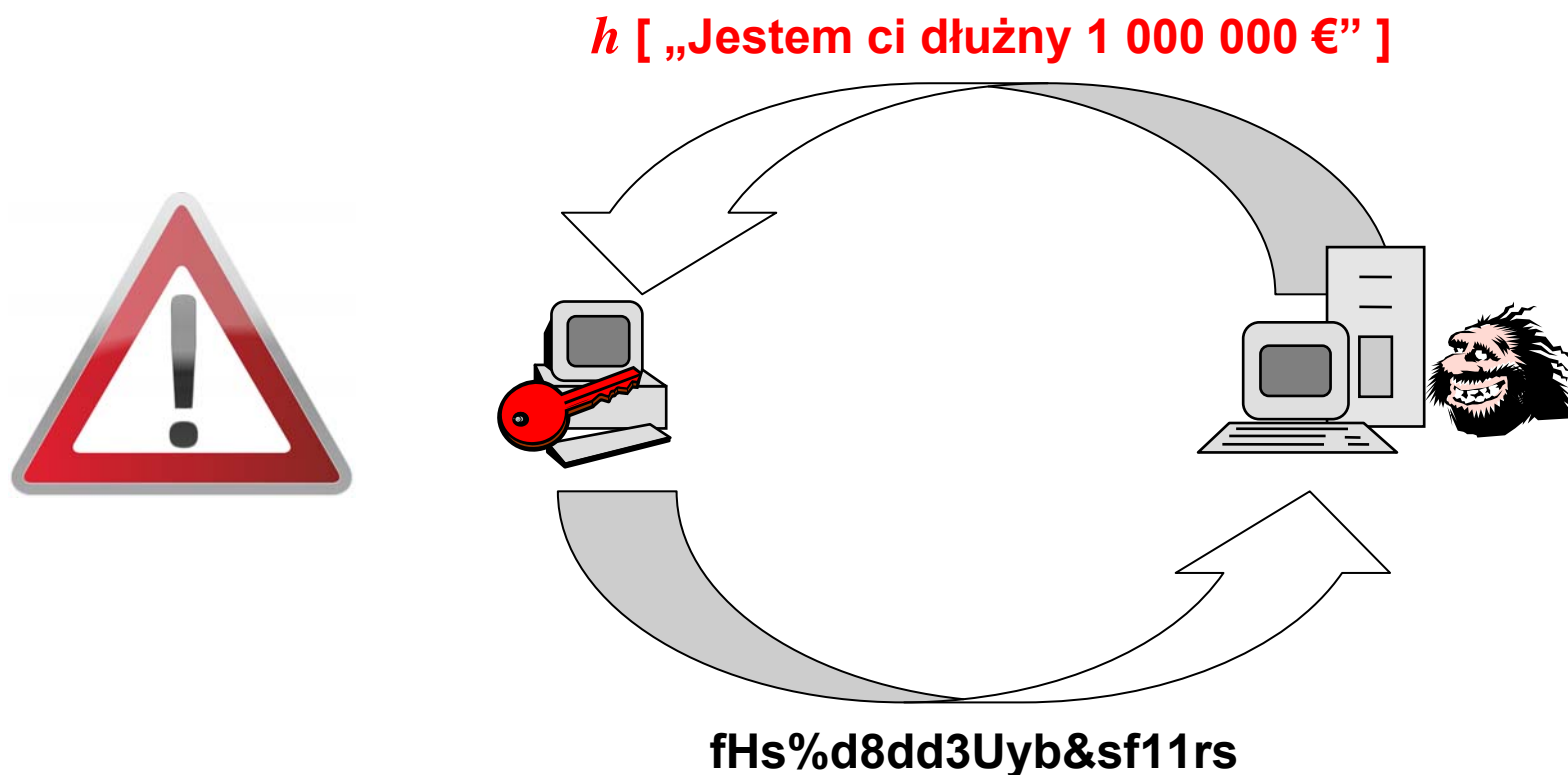
- serwer pyta o nazwę użytkownika, a następnie przesyła unikalny ciąg („zwołanie”)
- klient **szyfruje** (podpisuje) otrzymany ciąg **kluczem prywatnym** i odsyła „odzew”



Uwierzytelnianie

Uwaga:

- kryptografia asymetryczna jest wystarczająca do uwierzytelniania
- jednak stosowanie jej w takim prostym schemacie „zawołanie-odzew” rodzi pole do nadużyć:



Uwierzytelnianie

Web Authentication: An API for accessing Public Key Credentials Level 1



W3C Recommendation, 4 March 2019

§ 10.2. Simple Transaction Authorization Extension (txAuthSimple)

This extension allows for a simple form of transaction authorization. A [Relying Party](#) can specify a prompt string, intended for display on a trusted device on the authenticator.

Narzędzia



Trusted Platform Module (TPM)



por. smart cards

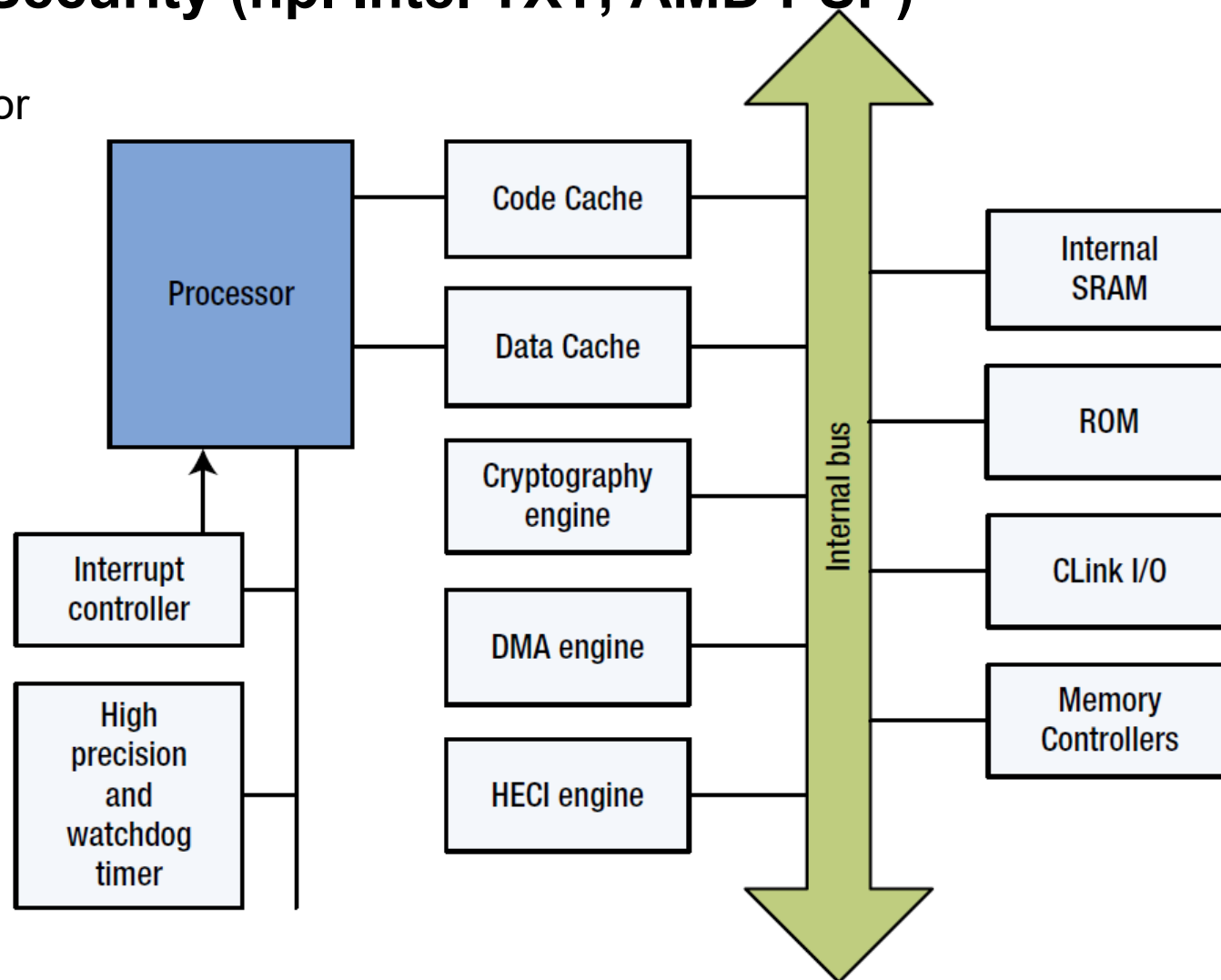
→ mmc tpm.msc

→ <https://resources.infosecinstitute.com/uefi-and-tpm/>

Narzędzia

Platform Embedded Security (np. Intel TXT, AMD PSP)

- chipset coprocessor
- TPM
- BootGuard
- crypto engine



Narzędzia

Hardware Security Module (HSM)



↔ PCSK#11



Narzędzia

Przykładowe biblioteki kryptograficzne

systemowe:

- Unix: libcrypt
- Windows: wincrypt
- .NET: System.Security.Cryptography

darmowe:

- cryptix www.cryptix.org
- crypto++ www.cryptopp.com
- Boucy Castle www.bouncycastle.org

komercyjne:

- RSA BSAFE www.rsasecurity.com
- cryptlib www.cryptlib.com
- nCiphers www.ncipher.com

AES-NI =

AES New Instructions set
w procesorach Intel oraz AMD

"Anyone who creates his or her own cryptographic primitives is either a genius or a fool. Given the genius/fool ratio for our species, the odds aren't very good."

Bruce Schneier



Znaczenie właściwej edukacji

bcrypt:	1999
PBKDF2:	2000
scrypt:	2009
Argon2:	2015

2019

How to encrypt password on client side using Javascript

```
document.getElementById("hide").value =  
document.getElementById("password").value;  
var hash = CryptoJS.MD5(pass);  
document.getElementById('password').value=hash;  
return true;
```

Przyszłość kryptografii

Przyszłość kryptografii

Ciekawe trendy:

- szyfry homomorficzne: możliwe przetwarzanie zaszyfrowanych danych bez konieczności uprzedniego rozszyfrowania: $D[E[M_1] \circ E[M_2]] = M_1 \diamond M_2$
- stateful encryption: dla ograniczonej mocy obliczeniowej (np. IoT)

Zwiększanie mocy obliczeniowej:

- inżynieria biogenetyczna – DNAE (***DNA Encryption***)
- procesory kwantowe (www.qubit.org)

Silniejsze szyfry:

- kryptografia kwantowa – QKD (***Quantum Key Distribution***)
- i post-kwantowa – np. Quantum-Resistant OTR (***Off-The-Record***) Messaging

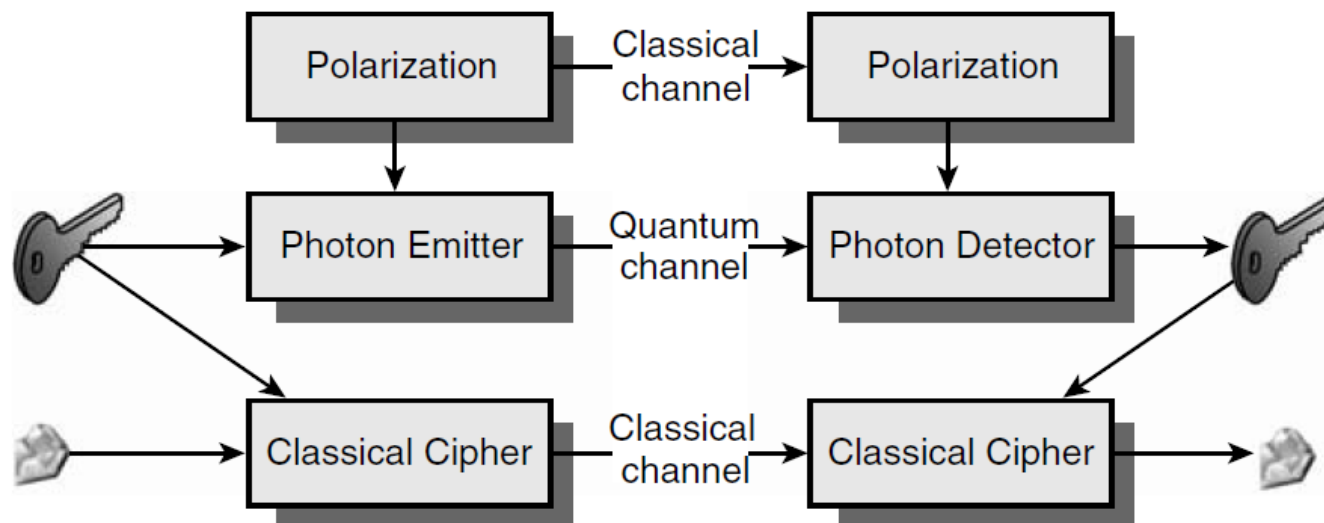
Przyszłość kryptografii

Algorithm :	Quantum Safe?	Usage :	Replace with:
Diffie-Hellman key agreement	No	Establishment of an ephemeral shared secret between Alice and Bob for AES	Quantum-safe KEX (fast enough to re-key often), such as R-LWE
AES-128 (CTR mode)	Somewhat (AES effective key sizes cut in half)	Encryption, once shared secret exists; Stream cipher (CTR mode), allows plausible forgeability by a third party, enabling repudiability goals to be met.	AES-256
128-bit truncated SHA-1	Somewhat	Computing hash of shared secret $g^{(x1y1)}$ to get enc key, $H(enc)$ to get MAC key	Something that allows for more bits of QS security for HMAC
MACs	Depends on underlying hard problem upon which MAC is based	Enables repudiable mutual authentication of Alice and Bob for each message	MAC based on a cryptographic problem that is still secure against a quantum adversary.
RSA Digital Signatures	No	Authentication of the <i>initial</i> Diffie-Hellman key agreement	Quantum-safe digital signature

Przyszłość kryptografii

Kryptografia kwantowa (np. BB84, E91, B92)

- Alice generates random key and encoding bases.
- Alice sends the polarized photons to Bob.
- Alice announces the polarization for each bit.
- Bob generates random encoding bases.
- Bob measures photons with random bases.
- Bob announces which bases are the same as Alices.



BB84 (M. Sosenkin, *Introduction to Quantum Cryptography*, Polytechnic University, New York 2005 [<http://sfs.poly.edu/presentations/MikeSpres.pdf>].)

Kryptografia

nie rozwiązuje wszystkich problemów

WSJ CYBERSECURITY

Capital One Breach Highlights Shortfalls of Encryption

Capital One said in a statement this week that it uses encryption “as a standard,” but the method used by the hacker “enabled the decrypting of data.” The bank didn’t respond to questions about its encryption practices.

Kryptografia

nie rozwiązuje wszystkich problemów

Krytyczna podatność w Windows. Można podrabiać podpis cyfrowy (w tym certyfikaty SSL/TLS).

14 STYCZNIA 2020, 19:39

” *An attacker could exploit the vulnerability by using a spoofed code-signing certificate to sign a malicious executable, making it appear the file was from a trusted, legitimate source. The user would have no way of knowing the file was malicious, because the digital signature would appear to be from a trusted provider.*

” *By exploiting this vulnerability, an attacker may be able to spoof a valid X.509 certificate chain on a vulnerable Windows system. This may allow various actions including, but not limited to, interception and modification of TLS-encrypted communications or spoofing an **Authenticode** signature.*

” *Will confirms all X.509 validation broken, not just code signing. Okay, I'm back on the hype train, that's pretty bad. <https://t.co/6rBV1lu4Yk>*
— Tavis Ormandy (@taviso) *January 14, 2020*

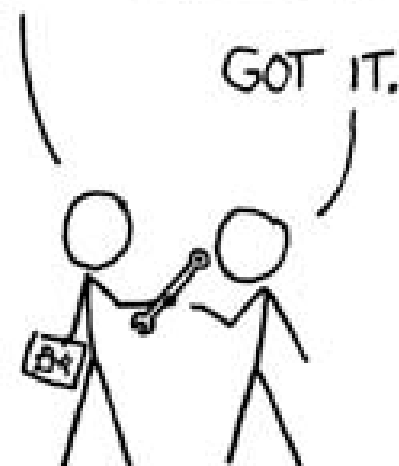
A CRYPTO NERD'S IMAGINATION:

HIS LAPTOP'S ENCRYPTED.
LET'S BUILD A MILLION-DOLLAR
CLUSTER TO CRACK IT.



WHAT WOULD ACTUALLY HAPPEN:

HIS LAPTOP'S ENCRYPTED.
DRUG HIM AND HIT HIM WITH
THIS \$5 WRENCH UNTIL
HE TELLS US THE PASSWORD.



<https://xkcd.com>



Steganografia

στεγανός (steganós) – niewidoczne
γράφειν (gráfein) – pismo

Jan Tritemiusz – „Steganographia, hoc est ars per occultam ...” XV/XVI w

Ukrywanie informacji wewnątrz większych zbiorów danych

- ochrona poufności danych niejawnych a nawet ukrycie faktu ich istnienia
- tworzenie zamaskowanych kanałów komunikacyjnych
- znaki wodne (*watermarking*) wykonywanie podpisów identyfikujących porcje danych (znaki wodne w grafice i strumieniach video) – ochrona praw autorskich

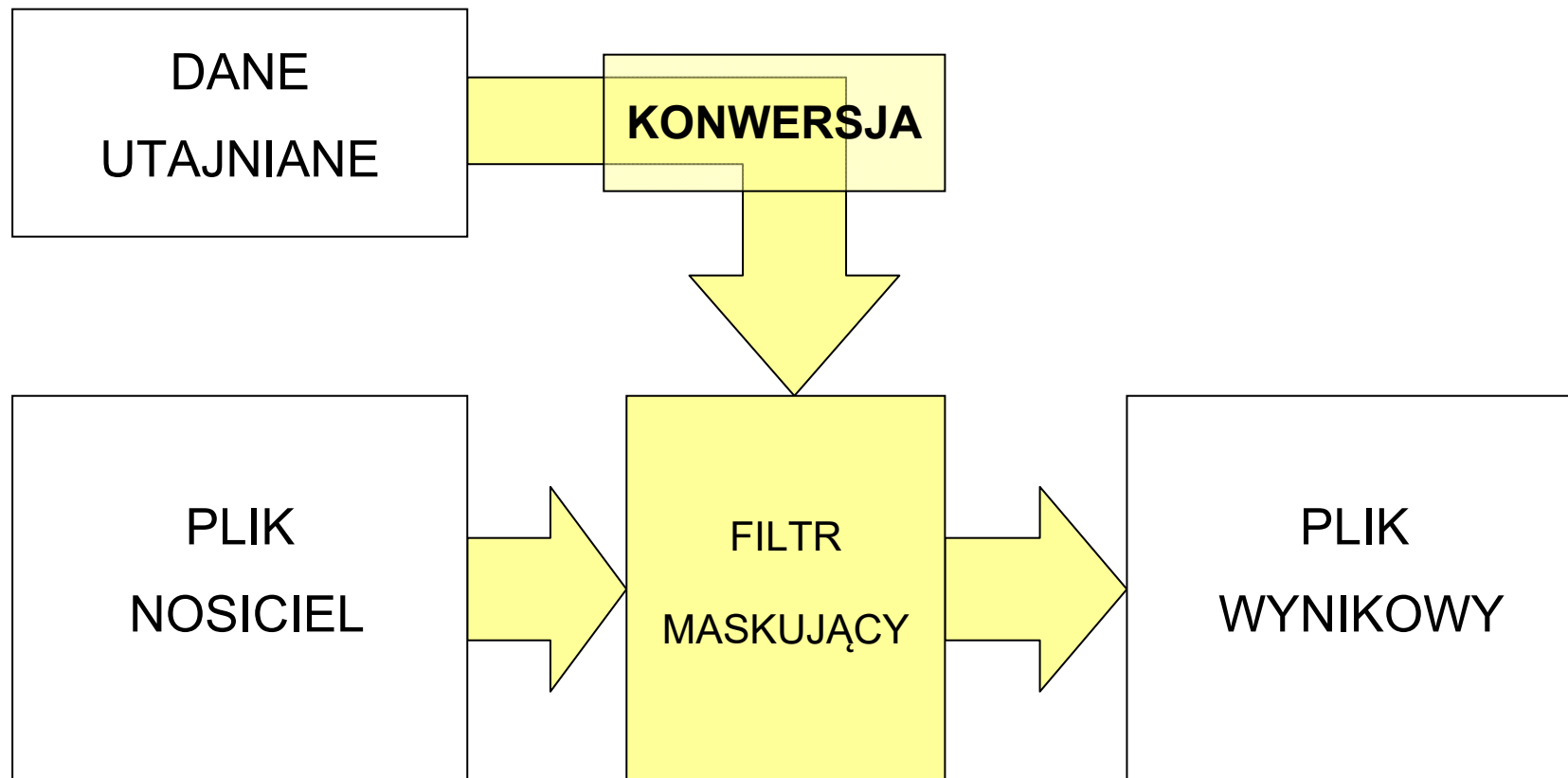
<http://www.securityfocus.com/infocus/1684>

<http://www.jjtc.com/stegdoc/steg1995.html>

<http://www.cotse.com/tools/stega.htm>

Steganografia

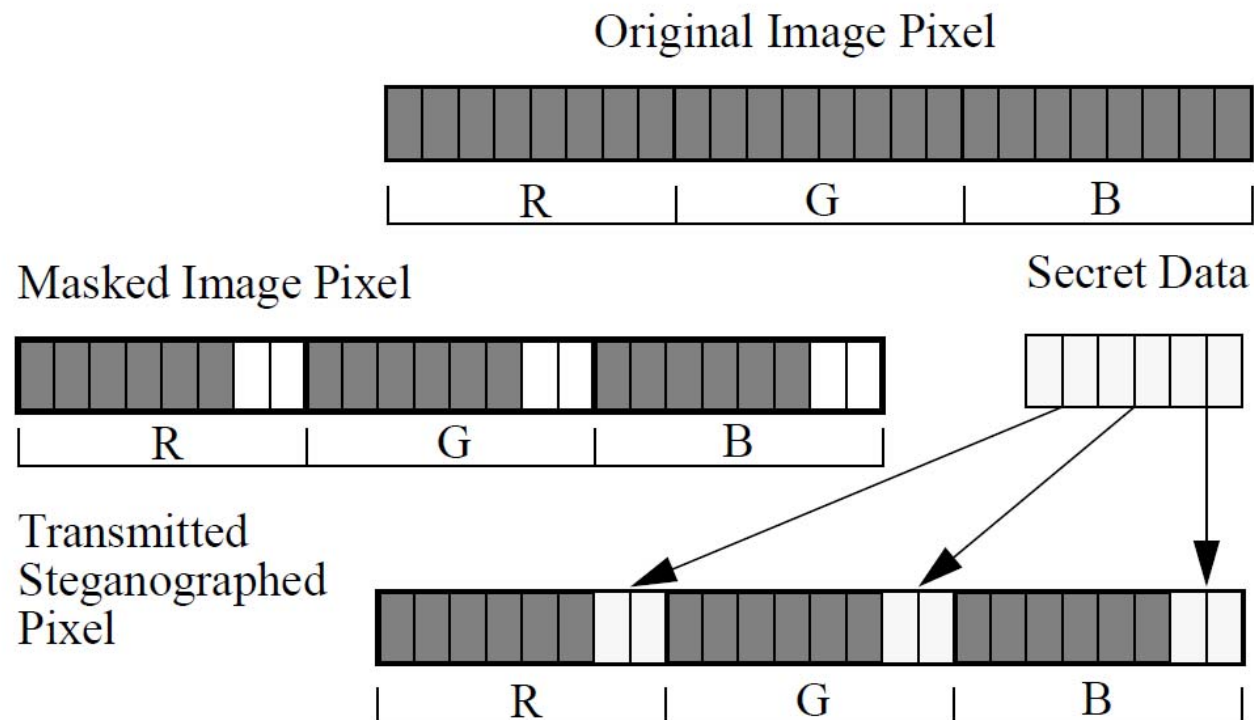
Schemat kodowania



Steganografia

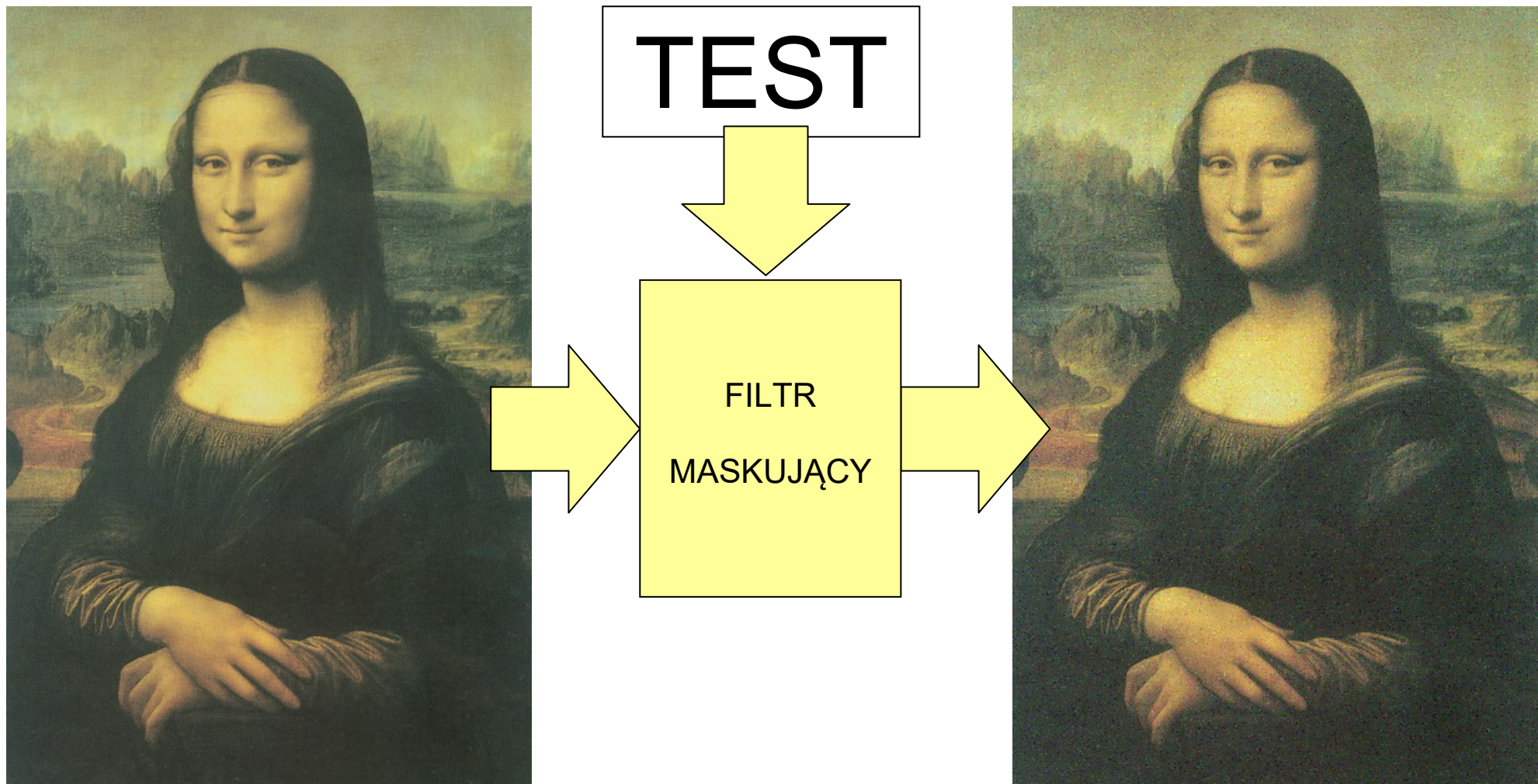
Grafika rastrowa

- szum mapy barwnej:



Steganografia

Przykład kodowania (znak wodny)



Steganografia

Przykład dekodowania

