ZASTOSOWANIE KRYPTOGRAFII

PIOTR DOBOSZ

SSL (SECURE SOCKETS LAYER)

- Opracowany i wdrożony przez Netscape Communications Corporation w 1994 r.
- Zabezpiecza komunikację w Internecie
- Wymaga certyfikatu
- Zapewnia poufność przesyłanych danych

SSL/TLS (TRANSPORT LAYER SECURITY)

- Internet Engineering Task Force (IETF) w 1996 roku tworzy grupę TLS mającą rozwijać standard SSL
- TLS 1.0 głównie zorientowany na uwierzytelnianie serwera (nadbudowa wersji SSL 3)
- Kolejne wersje (TLS 1.1 i 1.2) dodają nowe zalecenia, np. co do długości kluczy, zalecanych algorytmów oraz rozwiązują problemy poprzednich wersji; obecnie zainaugurowane zostały prace nad wersją 1.3 (2018)

ALGORYTM SZYFROWANIA SSL

- SSL wykorzystuje zarówno symetryczne jak i asymetryczne algorytmy
- Pozwala na wykorzystanie się niemal w każdym protokole wymiany komunikacji (Telnet, SMTP, POP3/IMAP)
- Najczęściej stosowane są algorytmy: DH (Diffiego-Hellmana), DES, 3DES, AES, RSA

PODPROTOKOŁY SSL

- SSL Handshake – definiuje metody negocjowania parametrów bezpiecznej sesji, czyli algorytmów szyfrowania danych, algorytmów uwierzytelniania i integralności informacji
- SSL Change Cipher – protokół zmiany specyfikacji szyfru SSL
- SSL Alert Protocol – protokół alarmowy SSL
- SSL Record – definiuje format przesyłanych pakietów danych

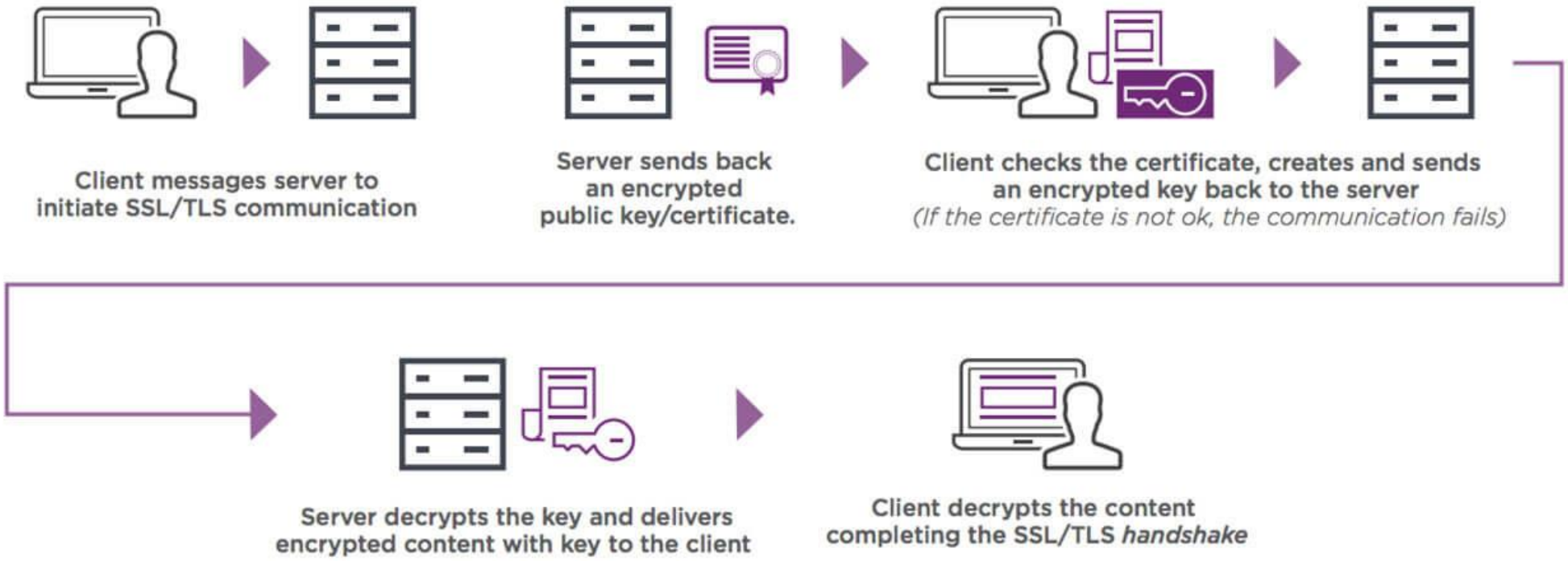
CERTYFIKAT SSL

- Nazwa domeny
- Okres ważności certyfikatu
- Szczegóły dotyczące urzędu certyfikacji (CA)
- Klucz publiczny i wersja SSL/TLS
- Algorytm klucza publicznego
- Algorytm podpisu certyfikatu

TYPY CERTYFIKATÓW

- DV (Domain Validation)- jest to najprostsza wersja certyfikatu SSL. Zabezpiecza ona transmisje danych w obrębie domeny oraz potwierdza jej autentyczność.
- OV (Organization Validation)- odpowiedni dla dużych sklepów internetowych, serwisów hotelowych czy też stron samorządowych.
- EV (Extended Validation)- Certyfikaty te zabezpieczają domenę, właściciela oraz wyświetlają zielony pasek startu. Używany jest do stron wrażliwych takich jak bankowość internetowa, czy strony posiadające delikatne informacje.

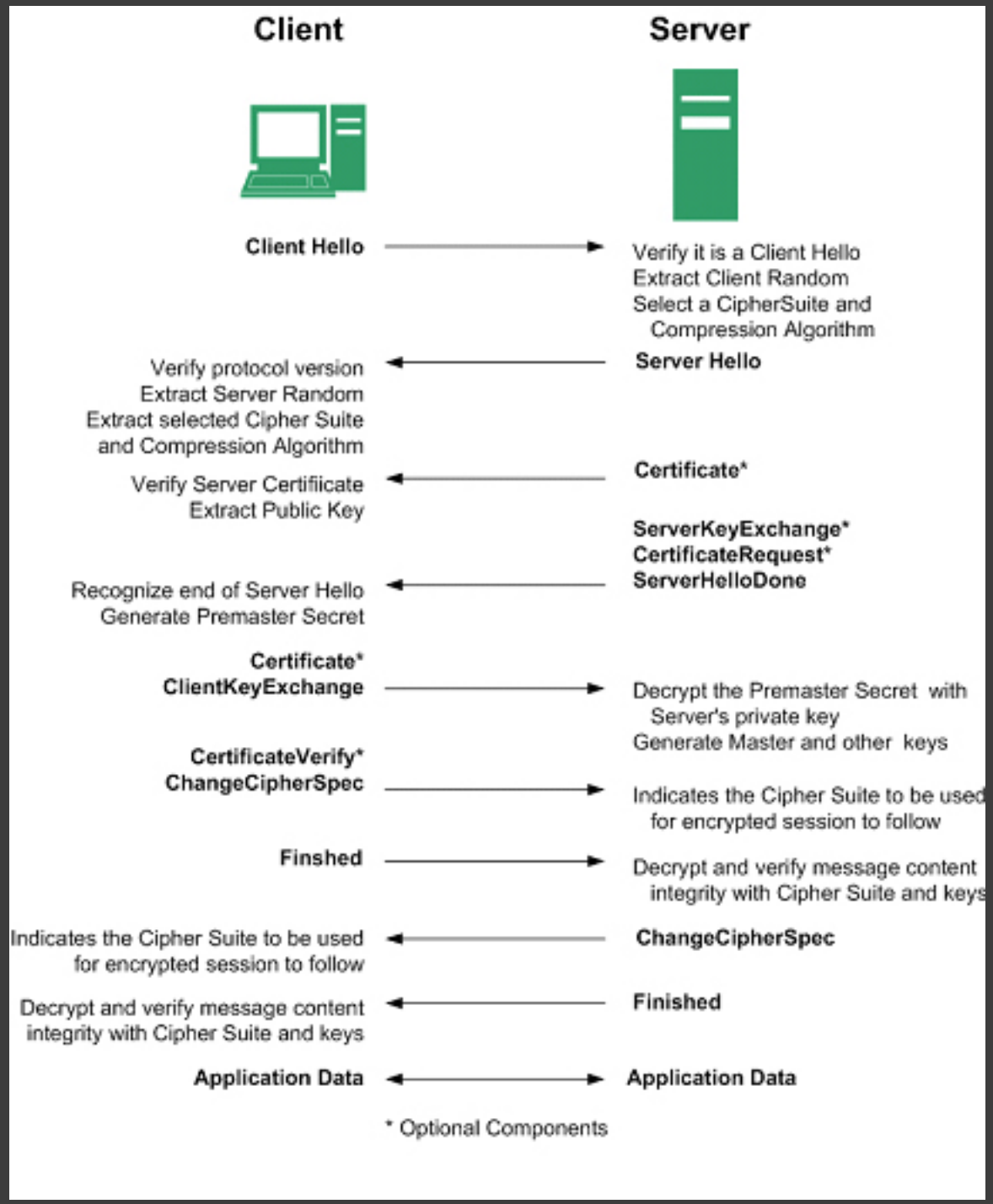
DZIAŁANIE SSL



https://www.entrust.com/-/media/entrust/resources/product-support/certificate-solutions/1258x489_how-ssl-certificates-work.jpg?la=en&hash=21C4AB8CF8CE8F9DAD70C19DC903DCB7

DZIAŁANIE SSL

- https://help.sonicwall.com/help/sw/eng/7030/25/9/0/content/images/SSL_Control_establishment.gif



SSH (SECURE SHELL)

- Grupa protokołów komunikacyjnych
- Służy bezpiecznemu strumieniowaniu danych pomiędzy dwoma węzłami
- Dba o integralność oraz bezpieczeństwo przesyłanych danych
- Dostępny zarówno jako narzędzie komercyjne (oryginalne) oraz w wersji otwartoźródłowej (OpenSSH)

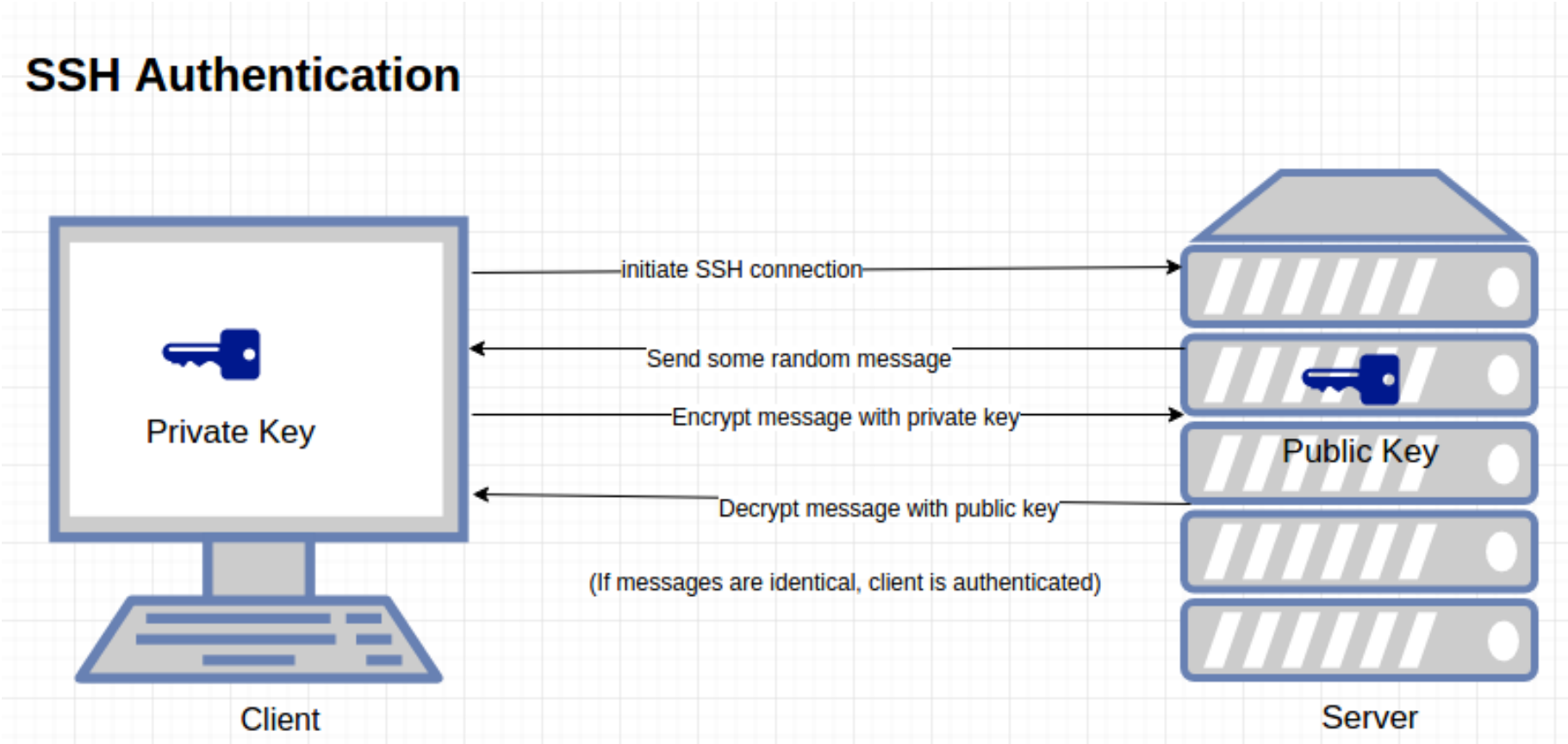
WERSJE SSH

- SSH-1, pierwsza wersja, opracowana w 1995 roku na Technicznym Uniwersytecie Helsińskim przez Tatu Ylönen
- Pierwotnie freeware z grupą entuzjastów, wspierających rozwój narzędzia; późniejsza ewolucja w zamknięte oprogramowanie
- SSH-2 – niekompatybilna wersja z SSH-1. Opracowywane przez IETF (Internet Engineering Task Force). Odpowiedź na podatności wersji pierwszej (wstrzykiwanie komend w zaszyfrowany strumień)
- Wersja druga używa protokołu Diffiego-Hellmana oraz szyfrowania AES. Dodatkowo pozwala na wykorzystywanie autoryzacji po kluczach (RSA) czy protokole Kerberos.

ZASTOSOWANIE SSH

- W głównej mierze – komunikacja z konsolą serwerową (zdalną)
- Przesyłanie plików na odległość
- Umożliwianie wykorzystania urządzeń w sieciach zdalnych (tunelowanie)
- Operacje zdalmne na systmie plików (sshfs)

DZIAŁANIE PROTOKOŁU



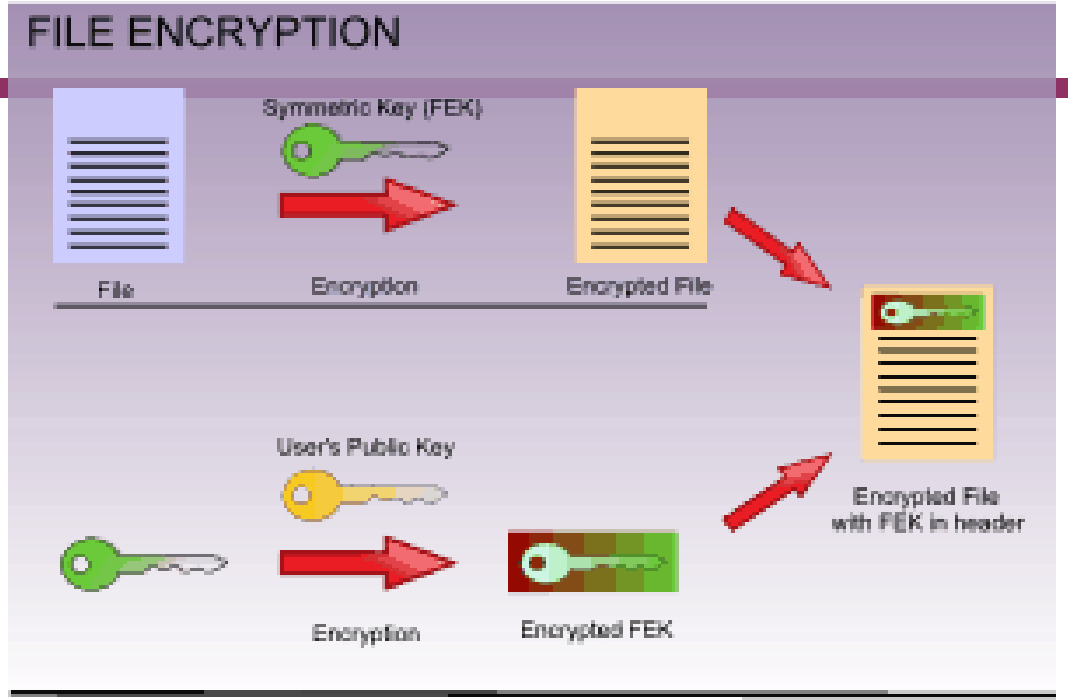
https://vandana759.files.wordpress.com/2017/06/sshauthentication.png

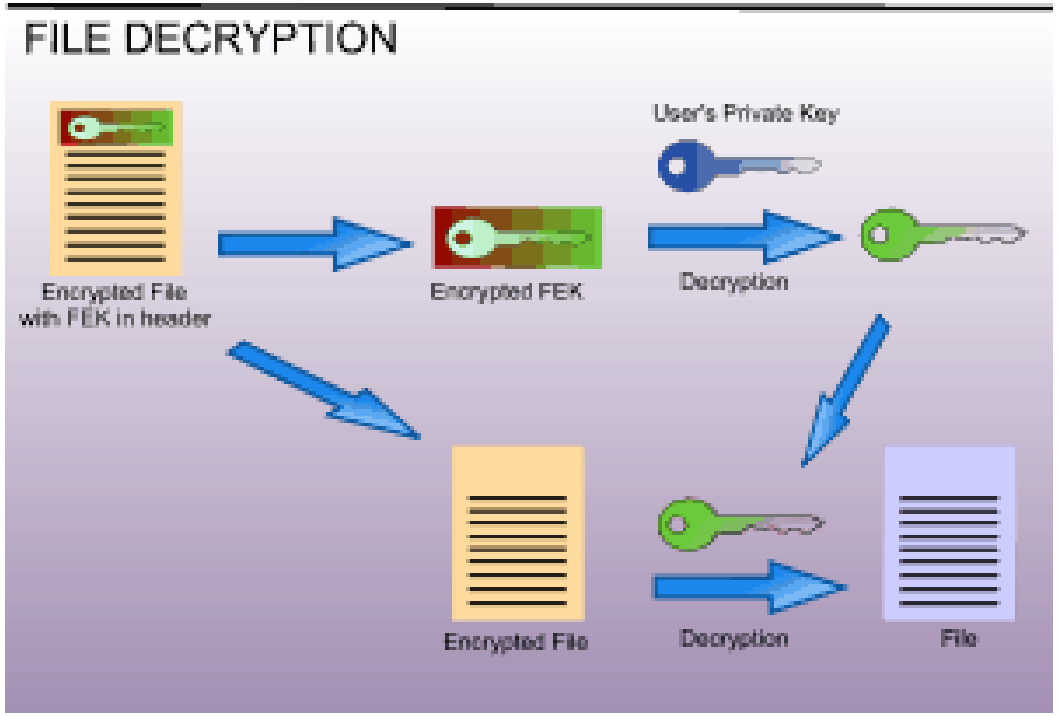
SZYFROWANIE PLIKÓW I KATALOGÓW

- Coraz większą wagę przykłada się do ochrony zapisanych danych na urządzeniach pamięci masowej
- Coraz częściej dochodzi do naruszenia prywatności danych, w tym domniemania pozyskania/usunięcia ważnych informacji
- Coraz większa potrzeba zabezpieczenia co ważniejszych danych

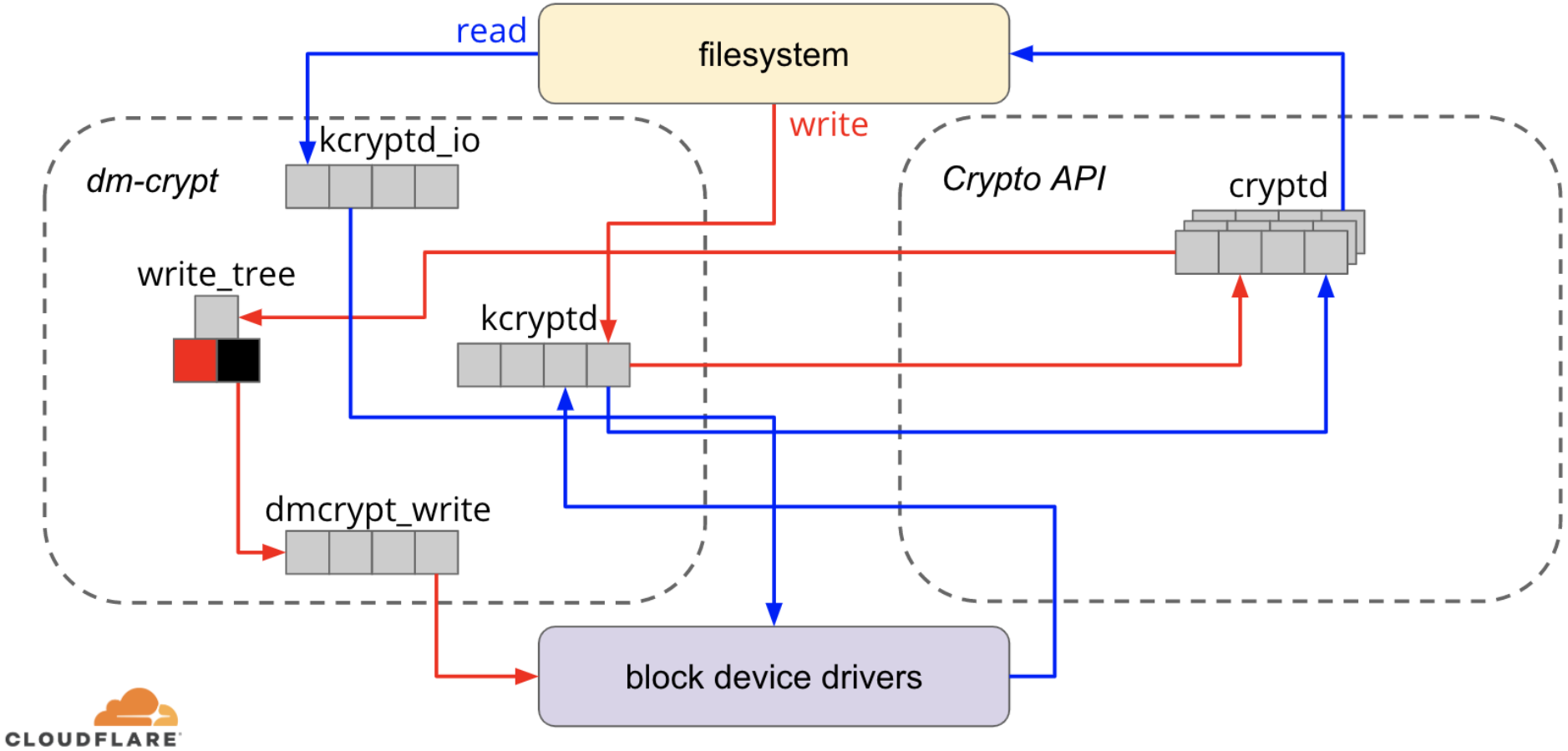
EFS

- https://www.google.com/url?sa=i&url=https%3A%2F%2Fnetworkencyclopedia.com%2Fencrypting-file-system-efs%2F&psig=AOvVaw0CPV_3P7wNqhOwGYGDW7um&ust=1610832917024000&source=images&cd=vfe&ved=0CAkQjhxqFwoTCMiskrDynu4CFQAAAAAdAAAAABAY





FSCRYPT



https://blog-cloudflare-com-assets.storage.googleapis.com/2020/03/dm-crypt.png

BITLOCKER

- https://images.slideplayer.com/42/11313417/slides/slide_6.jpg

AES-CBC + diffuser

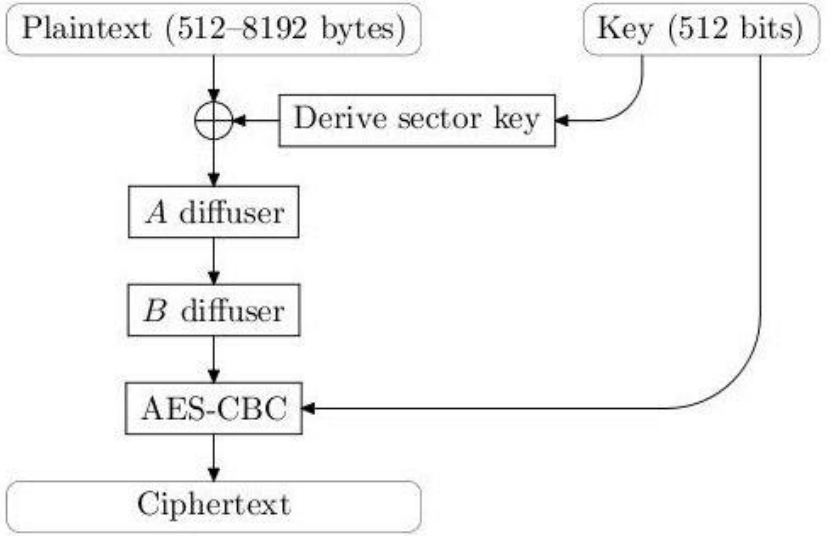
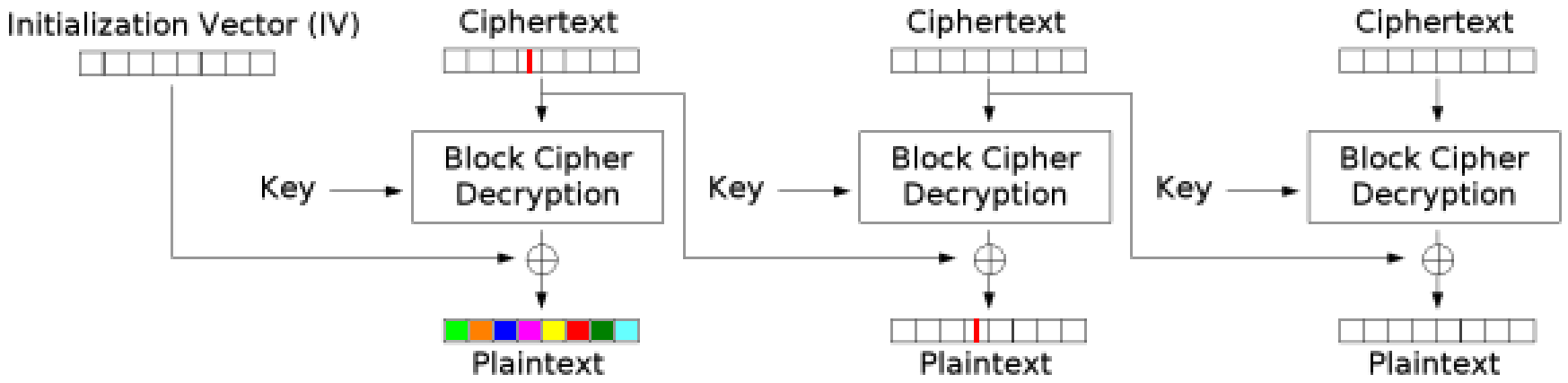


Figure 1: An overview of AES-CBC + diffuser

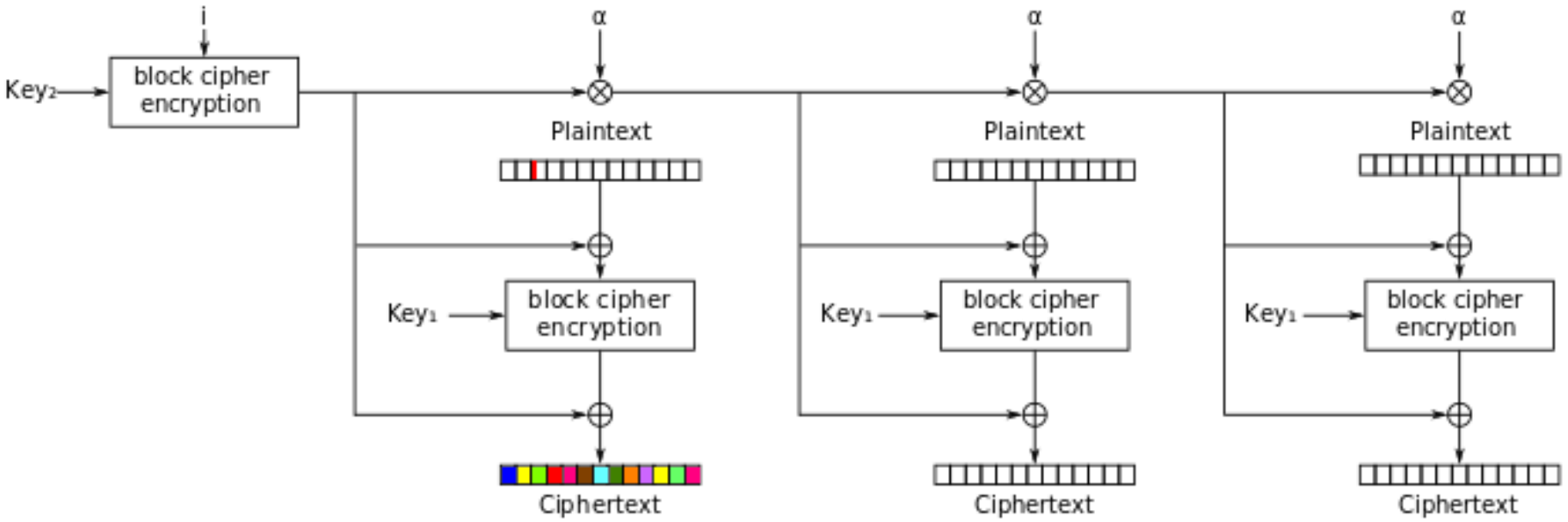
- Block size can be anything between 512-8192 (any power of 2)
- Plaintext is XORed with a sector key -> Plaintext runs through 2 un-keyed diffuser -> Plaintext is encrypted with AES-CBC
  - The sector key and the AES-CBC key are independent keys
  - (256 + 256) lower keys are possible, which means unused bits (128)

BITLOCKER CDC-AES



Cipher Block Chaining (CBC) mode decryption

BITLOCKER XST



XEX with tweak and ciphertext stealing (XTS) mode encryption

CBC VS XST

$$C_i = E_K(C_{i-1} \oplus P_i).$$

$$X = E_K(I) \otimes \alpha^j,$$
$$C = E_K(P \oplus X) \oplus X,$$

where:

P is the plaintext,
I is the number of the sector,
α is the primitive element of $\mathrm{GF}(2^{128})$ defined by polynomial x; i.e., the number *2*,
j is the number of the block within the sector.

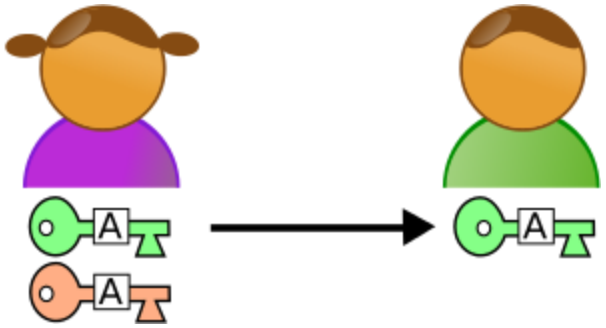
INNE ROZWIĄZANIA SZYFRUJĄCE DANE

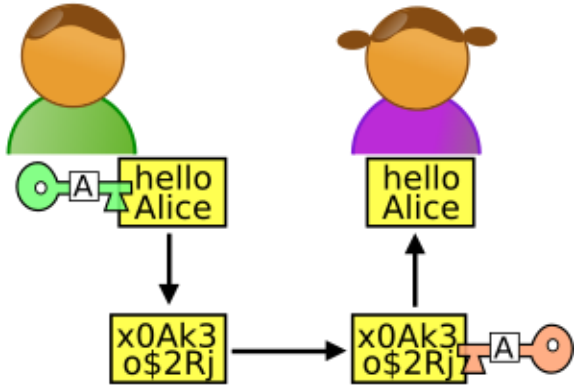
- TrueCrypt (niebiezpieczny!)
- VeraCrypt
- LUKS (cryptsetup)
- dm-crypt

KRYPTOGRAFIA ASYMETRYCZNA

- Składa się z co najmniej dwóch kluczy
- Jeden z kluczy możemy dystrybuować w nieograniczonej ilości
- Drugi (prywatny) musimy zachować w poufności, w mało dostępnym miejscu
- Całość opiera się na funkcjach jednokierunkowych
- Przykładowo potęgowanie modulo a logarytmowanie dyskretne

DZIAŁANIE KRYPTOGTAFII ASYMETRYCZNEJ





RSA (RIVEST-SHAMIR-ADLEMAN)

- Zaprojektowany w 1977
- Możliwy do wykorzystania przy szyfrowaniu i podpisie cyfrowym
- Operuje na dużych liczbach pierwszych; losowość oraz trudność procesu tzw. Faktoryzacji decyduje o zabezpieczeniach wdrażanego algorytmu

RSA - DZIAŁANIE

W celu wygenerowania pary kluczy (prywatnego i publicznego):

- wybrać losowo dwie duże liczby pierwsze p i q (najlepiej w taki sposób, aby obie miały zbliżoną długość w bitach, ale jednocześnie były od siebie odległe wartościami).
- Obliczyć wartość $n = pq$.
- Obliczyć wartość funkcji Eulera dla n: $\phi(n)=(p-1)(q-1)$
- Wybierać liczbę e ($1 < e < \phi(n)$) względnie pierwszą z $\phi(n)$.
- Znaleźć liczbę d, gdzie jej różnica z odwrotnością modularną liczby e jest podzielna przez $\phi(n)$:
- $d \equiv e^{-1} \pmod{\phi(n)}$.
- Ta liczba może być też prościej określona wzorem:
- $d \cdot e \equiv 1 \pmod{\phi(n)}$.

Klucz publiczny jest definiowany jako para liczb (n, e), natomiast kluczem prywatnym jest para (n, d).

RSA - PRZYKŁAD

Calculation of Modulus And Totient

Lets choose two primes: $p = 11$ and $q = 13$. Hence the modulus is $n = p \times q = 143$. The totient of n $\phi(n) = (p-1) \cdot (q-1) = 120$.

Key Generation

For the public key, a random prime number that has a greatest common divisor (gcd) of 1 with $\phi(n)$ and is less than $\phi(n)$ is chosen. Let's choose 7 (note: both 3 and 5 do not have a gcd of 1 with $\phi(n)$. So $e = 7$, and to determine d, the secret key, we need to find the inverse of 7 with $\phi(n)$. This can be done very easily and quickly with the *Extended Euclidean Algorithm*, and hence $d = 103$. This can be easily verified: $e \cdot d = 1 \bmod \phi(n)$ and $7 \cdot 103 = 721 = 1 \bmod 120$.

Encryption/Decryption

Lets choose our plaintext message, m to be 9:

Encryption:

$$m^e \bmod n = 9^7 \bmod 143 = 48 = c$$

Decryption:

$$c^d \bmod n = 48^{103} \bmod 143 = 9 = m$$

DIGITAL SIGNATURE ALGORITHM (DSA), ALGORYTM PODPISU CYFROWEGO

- Stworzony dla podpisów cyfrowych
- Wykorzystuje koncepcję modularnego potęgowania i dyskretnego problemu logarytmu
- Tak jak RSA – jest jednokierunkowy

GENEROWANIE

Generacja klucza

Generowanie klucza ma dwie fazy. Pierwsza faza to wybór *parametrów algorytmu*, które mogą być współdzielone przez różnych użytkowników systemu, podczas gdy druga faza polega na obliczeniu jednej pary kluczy dla jednego użytkownika.

Generowanie parametrów

- Wybierz zatwierdzoną kryptograficzną funkcję skrótu z wyjściowymi bitami długości . W oryginalnym DSS zawsze był SHA-1 , ale silniejsze funkcje skrótu SHA-2 są zatwierdzone do użytku w obecnym DSS. Jeśli jest większa niż długość modułu , używane są tylko skrajne lewe bity wyniku skrótu. $H|H|H|H|NN$
- Wybierz długość klucza . Pierwotny DSS ograniczał się do wielokrotności 64 między 512 a 1024 włącznie. NIST 800-57 zaleca długości 2048 (lub 3072) dla kluczy z okresami ważności zabezpieczeń wykraczającymi poza 2010 (lub 2030). LL
- Wybierz taką długość modułu , że i . FIPS 186-4 określa i ma jedną z wartości: (1024, 160), (2048, 224), (2048, 256) lub (3072, 256). $NN < LN \leq |H|LN$
- Wybierz -bitową liczbę pierwszą . Nq
- Wybierz -bitową liczbę pierwszą taką, że - 1 jest wielokrotnością . $Lppq$
- Wybierz liczbę całkowitą losowo z . $h\{2 \dots p-2\}$
- Oblicz . W rzadkich przypadkach spróbuj ponownie z innym . Powszechnie używany. To modularne potęgowanie można skutecznie obliczyć, nawet jeśli wartości są duże. $g := h^{(p-1)/q} \mod pg = 1hh = 2$

Parametry algorytmu są (, ,). Mogą być współdzielone przez różnych użytkowników systemu. pqg

Klucze dla użytkownika

Biorąc pod uwagę zestaw parametrów, druga faza oblicza parę kluczy dla pojedynczego użytkownika:

- Wybierz liczbę całkowitą losowo z . $x\{1 \dots q-1\}$
- Oblicz . $y := g^x \mod p$

x jest kluczem prywatnym i jest kluczem publicznym. y

https://pl.qaz.wiki/wiki/Digital_Signature_Algorithm

PRZYKŁAD GENEROWANIA

Example of DSA signing by Alice

In practice, q is a 160-bit prime and h is a 160-bit integer representative of the SHA-1 hash of the message M, but the algorithm still works for our small numbers.

INPUT: Domain parameters $(p = 283, q = 47, g = 60)$
INPUT: Alice's private key, $a = 24$
INPUT: Message M with message digest $h = \text{Hash}(M) = 41$.

1. Alice chooses a random $k = 15$ in the range $[1, q - 1]$
2. Alice computes $X = g^k \bmod p = 60^{15} \bmod 283 = 207$ and
   $r = X \bmod q = 207 \bmod 47 = 19$.
   $r \neq 0$ so continue.
3. Alice computes $k^{-1} \bmod q = 15^{-1} \bmod 47 = 22$.
4. Alice computes $h = \text{Hash}(M) = 41$.
5. Alice computes $s = k^{-1}(h + ar) \bmod q = 22(41 + 24 \cdot 19) \bmod 47 = 30$.
   $s \neq 0$ so continue.
6. Alice issues the message M and signature $(r, s) = (19, 30)$.

PRZYKŁAD GENEROWANIA

Example of DSA verification by Bob (or anyone)

INPUT: Domain parameters $(p = 283, q = 47, g = 60)$
INPUT: Alice's public key, $A = 158$
INPUT: Message M with message digest $h = \mathrm{Hash}(M) = 41$.
INPUT: Signature $(r, s) = (19, 30)$.

1. Bob verifies that $0 < r = 19 < 47$ and $0 < s = 30 < 47 \Rightarrow$ OK, so continue.
2. Bob computes $w = s^{-1} \bmod q = 30^{-1} \bmod 47 = 11$.
3. Bob computes $h = \mathrm{Hash}(M) = 41$.
4. Bob computes $u_1 = hw \bmod q = 41 \cdot 11 \bmod 47 = 28$ and
   $u_2 = rw \bmod q = 19 \cdot 11 \bmod 47 = 21$.
5. Bob computes $X = g^{u_1} A^{u_2} \bmod p = 60^{28} \cdot 158^{21} \bmod 283 = 106 \cdot 42 \bmod 283 = 207$ and
   $v = X \bmod q = 207 \bmod 47 = 19$.
6. Bob checks that $v = 19 = r$, so he accepts the signature.

PODATNOŚCI DSA

- entropia, tajność i niepowtarzalność losowej wartości podpisu są krytyczne.
- Do ujawnienia klucza prywatnego wystarczy dwukrotne użycie tej samej wartości (nawet z zachowaniem tajemnicy), użycie przewidywalnej wartości lub wyciek nawet kilku bitów w każdym z kilku podpisów .
- Ten problem dotyczy zarówno DSA, jak i ECDSA - w grudniu 2010 roku grupa nazywająca się fail0verflow ogłosiła odzyskanie klucza prywatnego ECDSA używanego przez Sony do podpisywania oprogramowania dla konsoli do gier PlayStation 3 .
- Problemowi temu można zapobiec, określając deterministycznie wyprowadzanie z klucza prywatnego i skrótu wiadomości, zgodnie z opisem w dokumencie RFC 6979 . Gwarantuje to, że jest inny dla każdego i nieprzewidywalny dla atakujących, którzy nie znają klucza prywatnego .
- Ponadto złośliwe implementacje DSA i ECDSA mogą zostać utworzone, jeśli zostanie to wybrane w celu podśrodkowego wycieku informacji za pośrednictwem podpisów.

PODPIS CYFROWY

Wiadomość jest podpisana w następujący sposób: m

- Wybierz liczbę całkowitą losowo z $k\{1 \dots q-1\}$
- Oblicz . W mało prawdopodobnym przypadku zacznij ponownie od innego losowego . $r := \left(g^k \bmod p\right) \bmod qr = 0k$
- Oblicz . W mało prawdopodobnym przypadku zacznij ponownie od innego losowego . $s := \left(k^{-1}\left(H(m) + xr\right)\right) \bmod qs = 0k$

Podpis jest (r, s)

ELGAMAL

- Oparty na problemie logarytmu dyskretnego w ciele liczb całkowitych modulo z dużej liczby pierwszej
- Tak jak RSA, pozwala zarówno na wykonanie podpisu, jak i zaszyfrować/odszyfrować dowolne dane

GENEROWANIE KLUCZA

Generowanie klucza: wybieramy dowolną liczbę pierwszą p, dowolny generator α podgrupy multiplikatywnej, tzn. taki element, którego rząd jest równy $p-1$, oraz dowolne k takie, że: $1 < k < p$. Liczymy β

$$\beta = \alpha^k \mod p,$$

co potrafimy zrobić szybko za pomocą potęgowania przez podnoszenie do kwadratu.

Gdyby α było dowolne to przykładowo dla:

$p = 41,\ \alpha = 14,\ k = 16$ otrzymalibyśmy:

$$\beta = 14^{16} \mod 41 = 1,$$

co spowodowałoby, że kryptosystem byłby bezużyteczny (gdyż przy szyfrowaniu zawsze otrzymywalibyśmy 1).

Następnie publikujemy (p, α, β) jako klucz publiczny i zachowujemy (p, α, β, k) jako klucz prywatny.

Szyfrowanie: mając do zaszyfrowania wiadomość m, przedstawiamy ją jako element grupy $[1 < m < p-1]$ wybieramy losowo liczbę x i liczymy (modulo p)

$$(\alpha^x, m \times \beta^x).$$

Deszyfrowanie: podnosimy otrzymane α^x do potęgi k:

$$(\alpha^x)^k = \alpha^{kx} = \left(\alpha^k\right)^x = \beta^x.$$

Następnie znajdujemy odwrotność β^x (nadal modulo p) rozszerzonym algorytmem Euklidesa:

$$\gamma\beta^x + \delta p = 1,$$
$$\gamma\beta^x \equiv 1 \mod p,$$
$$\gamma \equiv (\beta^x)^{-1} \mod p.$$

W końcu dzielimy $m \times \beta^x$ przez β^x, czyli mnożymy przez jej odwrotność – γ:

$$(m \times \beta^x) \times \gamma \equiv m \times (\beta^x \times \gamma) \equiv m \times 1 \equiv m \mod p.$$

PODPIS CYFROWY

Żeby wygenerować podpis wiadomości m, losujemy liczbę r i liczymy:

$y = \alpha^r$ **(mod p)**,
$s = (H(m) - ky)r^{-1}$ **(mod(p-1))**, gdzie H jest funkcją haszującą.

Podpisem jest para (y, s)

Żeby zweryfikować podpis, sprawdzamy równanie:

$$\beta^y y^s = \alpha^{H(m)}.$$

Dla prawidłowego podpisu będzie się zgadzać:

$$\alpha^{ky}\alpha^{rs} = \alpha^{H(m)},$$
$$\alpha^{ky+r\left((H(m)-ky)r^{-1}\right)} = \alpha^{H(m)},$$
$$\alpha^{ky+H(m)-ky} = \alpha^{H(m)},$$
$$\alpha^{H(m)} = \alpha^{H(m)}.$$

Ważne jest zachowanie tajności wylosowanego r. Jeśli r byłoby znane, to można by odzyskać klucz prywatny z podpisu:

$$y^{-1}\left(H(m) - sr\right) = y^{-1}\left(H(m) - (H(m) - ky)r^{-1}r\right) = y^{-1}ky = k.$$

CIPHER BLOCK CHAINING – CBC

Operacje wykorzystywane podczas szyfrowania wyglądają następująco:

$$C_0 = IV$$
$$C_i = E(P_i \oplus C_{i-1})$$

natomiast podczas deszyfrowania tak:

$$P_i = D(C_i) \oplus C_{i-1}$$

https://pl.wikipedia.org/wiki/Tryb_wiązania_bloków_zaszyfrowanych

CTR

- Wybierana jest jawna funkcja, która na podstawie pozycji bloku danych względem początku strumienia generuje ciąg danych o długości równej długości bloku szyfru blokowego; wymagane jest, by funkcja ta zwracała wartości pseudolosowe.
- Wartość funkcji przekształcana jest za pomocą szyfru blokowego.
- Wynik operacji jest składany z wiadomością za pomocą operacji XOR; w ten sposób uzyskiwany jest fragment szyfrogramu.

https://pl.wikipedia.org/wiki/CTR_(tryb_licznikowy)

ALGORYTM RABINA

- Stworzony do podpisów cyfrowych
- Zaprojektowany i zaprezentowany przez Michael Oser Rabin w roku 1978
- Podpis przypomina RSA z exponent e=2, co z kolei znacznie poprawia bezpieczeństwo bezpośrednio powiązane z faktoryzacją
- Nie jest jednak w pełni ustandaryzowany, co przekłada się na rzadsze wykorzystanie w stosunku do RSA

GENEROWANIE PODPISU

Public key

A public key is a pair of integers (n, b) with $0 \leq b < n$ and n odd.

Signature

A signature on a message m is a pair (u, x) of a k-bit string u and an integer x such that

$$x(x + b) \equiv H(m, u) \pmod{n}.$$

Private key

The private key for a public key (n, b) is the secret odd prime factorization $p \cdot q$ of n, chosen uniformly at random from some space of large primes. Let $d = (b/2) \bmod n$, $d_p = (b/2) \bmod p$, and $d_q = (b/2) \bmod q$. To make a signature on a message m, the signer picks a k-bit string u uniformly at random, and computes $c := H(m, u)$. If $c + d^2$ is a quadratic nonresidue modulo n, then the signer throws away u and tries again. Otherwise, the signer computes

$$x_p := \left(-d_p \pm \sqrt{c + {d_p}^2} \right) \bmod p$$
$$x_q := \left(-d_q \pm \sqrt{c + {d_q}^2} \right) \bmod q,$$

using a standard algorithm for computing square roots modulo a prime—picking $p \equiv q \equiv 3 \pmod{4}$ makes it easiest. Square roots are not unique, and different variants of the signature scheme make different choices of square root;[4] in any case, the signer must ensure not to reveal two different roots for the same hash c. The signer then uses the Chinese remainder theorem to solve the system

$$x \equiv x_q \pmod{n}$$
$$x \equiv x_p \pmod{n}$$

for x. The signer finally reveals (u, x).

Correctness of the signing procedure follows by evaluating $x(x + b) - H(m, u)$ modulo p and q with x as constructed. For example, in the simple case where $b = 0$, x is simply a square root of $H(m, u)$ modulo n. The number of trials for u is geometrically distributed with expectation around 4, because about 1/4 of all integers are quadratic residues modulo n.

MATERIAŁY I ŹRÓDŁA

- https://jchost.pl/blog/ssl/
- https://mfiles.pl/pl/index.php/TLS
- https://pl.wikipedia.org/wiki/Transport_Layer_Security
- https://sites.google.com/site/tlsssloverview/ssl-tls-protocol-layers/record-layer
- https://www.ssh.com/ssh/protocol/
- https://panel.kylos.pl/knowledgebase/75/Co-to-jest-SSH-i-do-czego-mozna-go-wykorzystac.html
- https://www.ovh.pl/hosting/ssh.xml
- https://pl.wikipedia.org/wiki/Secure_Shell
- https://cryptoservices.github.io/fde/2014/12/08/code-execution-in-spite-of-bitlocker.html
- https://en.wikipedia.org/wiki/Disk_encryption_theory

MATERIAŁY I ŹRÓDŁA

- https://www.dobreprogramy.pl/BitLocker-bez-tajemnic,News,11349.html
- http://doctrina.org/How-RSA-Works-With-Examples.html
- https://pl.wikipedia.org/wiki/RSA_(kryptografia)
- https://asecuritysite.com/encryption/elgamal
- https://pl.wikipedia.org/wiki/ElGamal
- https://en.wikipedia.org/wiki/Rabin_signature_algorithm