Kryptografia asymetryczna

Kryptologia klucza jawnego

Szyfrowanie asymetryczne

- Generowana jest **para kluczy**
- Jeden z kluczy jest znany tylko właścicielowi (**klucz prywatny**) !!!
- Drugi jest znany publicznie (**klucz jawny**)
- Algorytmy szyfrowania i deszyfrowania są identyczne, przy czym używa się różnych kluczy z pary

Szyfrowanie asymetryczne

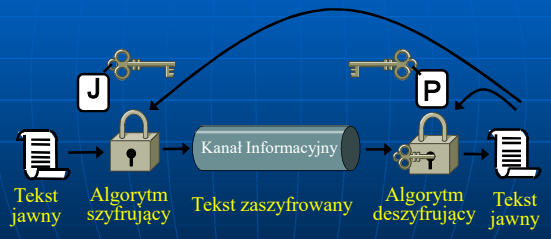
- Wiadomość zaszyfrowana kluczem jawnym może być odszyfrowana wyłącznie kluczem prywatnym
- Wiadomość zaszyfrowana kluczem prywatnym może być odszyfrowana wyłącznie kluczem jawnym

Szyfrowanie asymetryczne

Jeżeli zaszyfrujemy wiadomość kluczem jawnym, to:

- Tylko właściciel klucza prywatnego może wiadomość odszyfrować (bo nikt inny klucza prywatnego nie zna)
- Każdy może utworzyć taką zaszyfrowaną wiadomość (bo każdy może poznać klucz jawny)

Szyfrowanie asymetryczne 1

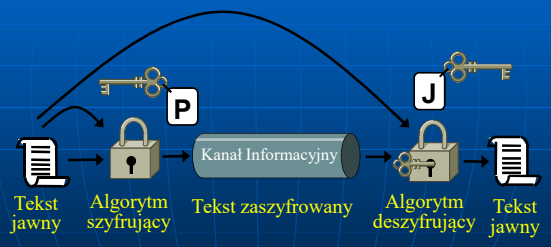


Szyfrowanie asymetryczne

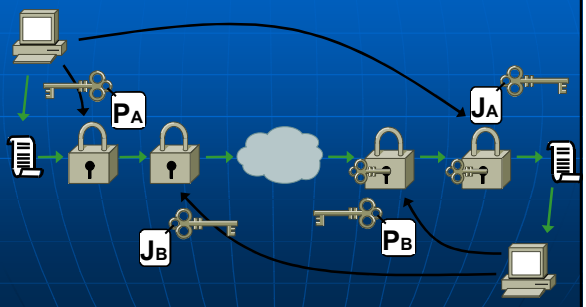
Jeżeli zaszyfrujemy wiadomość kluczem prywatnym, to:

- Każdy może wiadomość odszyfrować (kluczem jawnym)
- Jeżeli wiadomość została prawidłowo odszyfrowana kluczem jawnym, to mamy pewność, że została zaszyfrowana przez właściciela (bo nikt inny klucza prywatnego nie zna)

Szyfrowanie asymetryczne 2



Poufność, integralność, uwierzytelnianie i niezaprzeczalność – czy to możliwe ?



Szyfrowanie asymetryczne - zalety

- Ułatwiona dystrybucja kluczy:
  - klucz prywatny nigdy nie jest przesyłany
  - Klucz jawny jest jawny, nie ma problemu poufności podczas dystrybucji
- Silne i łatwe uwierzytelnianie (klucz prywatny)
- Niezaprzeczalność (klucz prywatny)

Symetryczne vs Asymetryczne

Szyfrowanie symetryczne	Szyfrowanie asymetryczne
Niezbędne do działania	
1. Ten sam algorytm i ten sam klucz używany jest do szyfrowania i deszyfrowania.	1. Ten sam algorytm używany do szyfrowania i deszyfrowania, para kluczy (szyfrowanie <-> deszyfrowanie).
2. Nadawca i odbiorca muszą mieć taki sam algorytm i klucz.	2. Nadawca i odbiorca muszą mieć algorytm oraz jeden z pary kluczy.

Symetryczne vs Asymetryczne

Szyfrowanie symetryczne	Szyfrowanie asymetryczne
Niezbędne do bezpieczeństwa	
1. Klucz nie może być ujawniony.	1. Jeden z kluczy nie może być ujawniony.
2. Odszyfrowanie komunikatu bez posiadania innych danych musi być niemożliwe lub zbyt kosztowne.	2. Odszyfrowywanie komunikatu bez posiadania innych danych musi być niemożliwe lub zbyt kosztowne.
3. Znajomość algorytmu oraz próbki tekstu zaszyfrowanego nie mogą być wystarczające do odkrycia klucza.	3. Znajomość algorytmu, jednego klucza i próbki tekstu zaszyfrowanego nie mogą być wystarczające do odkrycia drugiego klucza.

Wymagania dla systemów asymetrycznych

1. Strona B może łatwo wygenerować na drodze obliczeń klucz jawny i prywatny.
2. Nadawca A, znając klucz jawny B i tekst jawny może łatwo na drodze obliczeń stworzyć tekst zaszyfrowany.
3. Odbiorca B może łatwo otrzymać tekst jawny z tekstu zaszyfrowanego znając swój klucz prywatny.
4. Funkcje szyfrowania i deszyfrowania mogą być stosowane w dowolnej kolejności.

Wymagania dla systemów asymetrycznych – c.d.

5. Dla przeciwnika, znającego klucz jawny, określenie klucza prywatnego powinno być niewykonalne.
6. Dla przeciwnika, znającego klucz jawny i tekst zaszyfrowany określenie tekstu jawnego powinno być niewykonalne.

Algorytmy asymetryczne

- Algorytmy asymetryczne z kluczem publicznym opierają się na funkcjach matematycznych, a nie na podstawianiu i permutacji
- Są to algorytmy blokowe, z tym że bloki są traktowane jako liczby całkowite, a nie ciągi bitów

Algorytmy asymetryczne

- RSA
- ECDSA
- El-Gamal
- LUC
- DSA

➢ Stosowane długości kluczy:
1024, 2048, 4096, ... bitów

➢ Liczba 512-bitowa ma ok. 150 cyfr w systemie dziesiętnym

Algorytm RSA

- Generujemy klucze: *KJ = {e,n}*, *KP = {d,n}*
- Tekst dzielimy na bloki, bloki odczytujemy jako liczby całkowite *M* z przedziału *0..n-1*
- Szyfrowanie $C = M^e \bmod n$
- Deszyfrowanie $M = C^d \bmod n$

Generowanie kluczy RSA

p, q - liczby pierwsze	(prywatne, wybrane)
$n = pq$	(jawne, obliczone)
d takie że: $nwd(\Phi(n), d) = 1$, $1 < d < \Phi(n)$	(prywatne, obliczone)
$e \equiv d^{-1} \bmod \Phi(n)$	(jawne, obliczone)
$\{e,n\}$ klucz jawny	(jawne, obliczone)
$\{d,n\}$ klucz prywatny	(prywatne, obliczone)

RSA - trywialny przykład

1. Wybieramy liczby pierwsze *p*=7, *q*=17.
2. Obliczamy $n = 7 * 17 = 119$, $\Phi(n)=96$.
3. Wybieramy takie *e*, że *e* i *Φ(n)* są względnie pierwsze i *e*<*Φ(n)*. Stąd $e = 5$.
4. Obliczamy *d* takie, że $de \bmod 96 = 1$ i $d < 96$. Stąd $d = 77$, gdyż: $77 * 5 = 385 = 4 * 96 + 1$.
5. *KJ* = { 5, 119 }, *KP* = { 77, 119 }.

RSA - trywialny przykład - c.d.

1. KJ = { 5, 119 }, KP = { 77, 119 }.
2. Wartość tekstu jawnego M=19.
3. Szyfrowanie:
   $C = 19^5 \bmod 119 =$
   $2\,476\,099 \bmod 119 = 66.$
4. Deszyfrowanie:
   $M = 66^{77} \bmod 119 =$
   $1{,}27.. * 10^{140} \bmod 119 = 19.$

Aspekty obliczeniowe RSA

- Przy potęgowaniu uzyskujemy bardzo duże liczby, co utrudnia operacje modulo. Można skorzystać z własności arytmetyki modulo:

$$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$$

- Wykorzystuje się algorytmy przyspieszające potęgowanie.

Algorytm LUC

- Opracowany przez naukowców z Nowej Zelandii w 1993 roku
- Nie jest ograniczony umowami patentowymi jak to mam miejsce w przypadku RSA
- Zasada działania LUC opiera się na dużych liczbach całkowitych ciągu Lucasa.

Algorytm ElGamal

- 1985, ElGamal (Egipcjanin)
- Bazuje na problemie logarytmów dyskretnych
- RSA i ElGamal to najpopularniejsze obecnie algorytmy asymetryczne
- Poziom bezpieczeństwa wystarczający, gdy wybierana wartość modułu ma więcej niż 200 cyfr (600 bitów)

Klucze ElGamal

- Wybieramy:
  - dużą liczbę pierwszą *p* (moduł)
  - generator grupy cyklicznej *g*
  - liczbę całkowitą *k<p-1* (wykładnik)
- *Wyliczamy:* $q = g^k \bmod p$
- *KJ = { q, p, g }, KP = { q, p, g, k }*

Szyfrowanie ElGamal

Szyfrowanie

- Wybieramy wartość *s* (wykładnik)
- $C = (c_1, c_2)$, $c_1 = Mq^s$, $c_2 \equiv g^s \bmod p$

Deszyfrowanie

- Wyliczamy $(g^k)^s = c_2^k \equiv g^{sk} \bmod p$
- Wyliczamy odwrotność g^{-sk}
- $M = c_1 g^{-sk} \bmod p$

Kryptografia krzywych eliptycznych

$$y^2=x^3+ax+b$$

- Obliczenie dyskretnego algorytmu losowo wybranego elementu krzywej eliptycznej na podstawie znanego punktu bazowego krzywej jest niemożliwe.
- Bezpieczeństwo podobne do RSA przy kilkukrotnie krótszym kluczu
- Algorytm: ECDSA

Algorytmy asymetryczne

- Zapewniają większe bezpieczeństwo od algorytmów symetrycznych
- Czas obliczeń (szyfrowania) jest wielokrotnie dłuższy niż w przypadku algorytmów symetrycznych, np.:

$$C = 283^{938\,475} \bmod 4984$$

Asymetryczne - podsumowanie

- Generowana jest para kluczy: klucz prywatny i klucz jawny
- Algorytmy oparte na funkcjach matematycznych
- Bloki i klucze jako liczby całkowite
- Większe bezpieczeństwo
- Wielokrotnie dłuższe szyfrowanie
- Łatwa dystrybucja klucza (jawnego)

Kryptoanaliza algorytmów asymetrycznych

- Atak metodą brutalną
- Atak na podstawie klucza jawnego – próba wyliczenia klucza prywatnego na podstawie klucza jawnego
- Atak prawdopodobnego komunikatu – wszystkie możliwe komunikaty są szyfrowane kluczem jawnym i porównywane z tekstem zaszyfrowanym

Szyfrowanie asymetryczne

Które z usług ochrony są realizowane przez system szyfrujący asymetryczny?

- Poufność TAK
- Integralność TAK ... ale nie zawsze
- Uwierzytelnianie TAK
- Niezaprzeczalność TAK

Mieszanie

Funkcje hashujące
Kryptograficzne sumy kontrolne

Podpis cyfrowy

Co oznacza podpis na dokumencie ?

- Dokument jest oryginalny, nikt inny nie mógł złożyć tego podpisu (uwierzytelnienie)
- Dokument nie został zmodyfikowany (integralność)
- Osoba, która go podpisała nie może się tego wyprzeć (niezaprzeczalność)

Sygnatura cyfrowa - wymagania

- Musi umożliwiać sprawdzenie autora, daty i czasu sygnatury.
- Musi umożliwiać uwierzytelnienie w momencie podpisania.
- Musi umożliwiać weryfikację przez osoby trzecie w razie potrzeby rozstrzygnięcia konfliktu.

Podpis cyfrowy

Jak uzyskać najprostszy podpis cyfrowy ?

Zaszyfrować dokument swoim kluczem prywatnym.

Czy to wygodne rozwiązanie ?
I dlaczego nie ?

Jak zapewnić integralność nie sformatowanych danych

- Bit parzystości
- Cyfra kontrolna
- Suma kontrolna (CRC)
- …

Czy to wystarczy do bezpieczeństwa, gdy przeciwnikiem jest kryptoanalityk ?

Jednokierunkowe funkcje hashujące

Pierwotne zastosowanie - bazy danych (hashowanie):

1. Przy wpisywaniu danych (np. nazwisk) obliczamy sumę przez dodanie modulo 256 kolejnych bajtów nazwiska
2. Dla każdej z 256 sum tworzymy listę odpowiadających rekordów
3. Szukając danej osoby przeszukujemy tylko 1/256 rekordów w bazie

Własności funkcji hashującej

- Na podstawie danych dowolnej długości generuje krótką wartość o stałej długości (wiersze w tabeli wartości hashujących są wówczas podobnej długości)
- Dla różnych danych wartości funkcji powinny być z dużym prawdopodobieństwem różne

Funkcje hashujące w kryptografii i kodowaniu

- Mogą być wykorzystane do badania integralności danych
- Wynik działania funkcji nazywamy *wyciągiem* lub *skrótem*

Funkcje hashujące w kryptografii - wymagania

- Dla znanej wartości funkcji hashującej niemożliwe jest wyliczenie ciągu danych, z których wygenerowano skrót – funkcje jednokierunkowe
- Mając dany ciąg bajtów nie jest możliwe znalezienie drugiego ciągu o tej samej wartości hashującej.
  Taka para jest nazywana konfliktem

Jednokierunkowe funkcje hashujące - synonimy

- Funkcje kompresujące
- Funkcje koncentrujące
- Skróty wiadomości
- Funkcje skrótu
- MIC (*message integrity checks*)

Jednokierunkowe funkcje hashujące

- W typowych funkcjach hashujących nie wykorzystuje się tajnych kluczy – mogą być wygenerowane przez każdego
- Istnieją odwracalne funkcje hashujące z tajnym kluczem – kryptograficzne sumy kontrolne MAC (*message authentication code*)

Funkcja hashująca

- Z danych o zmiennym rozmiarze wylicza pewien wynik *H(M)* o stałym rozmiarze, zwany też wyciągiem lub skrótem komunikatu.
- Wynik hashowania jest funkcją wszystkich bitów komunikatu i zapewnia wykrywanie błędów

Hashowanie - wymagania

- *H* można zastosować do dowolnej wielkości bloku danych.
- *H* tworzy dane wyjściowe o ustalonej długości.
- *H(x)* łatwo obliczyć dla każdego *x*, co ułatwia implementację sprzętową i programową.
- Dla każdego kodu *m* znalezienie takiego *x*, że *H(x)=m* nie jest wykonywalne na drodze obliczeń.

Hashowanie - wymagania

- Dla każdego danego bloku x, znalezienie takiego y różnego od x, dla którego $H(y)=H(x)$ nie jest wykonywalne na drodze obliczeń.
- Znalezienie pary (x,y), że $H(y)=H(x)$ nie jest wykonywalne na drodze obliczeń

Funkcje hashujące

- MD5 (1991) – 128 bitów, znane ataki, nie powinna być stosowana
- SHA-1 (1994) – 160 bitów, znane ataki o złożoności 2^{63}, nie powinna być stosowana
- SHA-2 (2001) – 256, 384 lub 512 bitów, brak znanych ataków, zalecana obecnie
- SHA-3 (2012), wyłoniona w konkursie, na razie rzadko stosowana
- Inne: RIPE-MD160, MD4, MD2, N-Hash, Snerfu – bardzo rzadko używane

Ataki na funkcje hashujące

- Odwracanie funkcji (na razie udane tylko dla jednego z wariantów MD4)
- Znajdowanie konfliktów
  - Użyteczne są konflikty w przypadku znalezienia dwóch sensownych tekstów generujących ten sam skrót

Atak metodą „paradoksu dnia urodzin”

- Ile osób musi znajdować się na sali, aby prawdopodobieństwo że co najmniej dwie obchodzą urodziny tego samego dnia było większe od 0,5 ?
- Wystarczą 23 osoby !
- Tym samym, wystarczy wygenerować $2^{n/2}$ wiadomości, aby z prawdopodobieństwem ponad 50% znaleźć konflikt

Kryptograficzna suma kontrolna (MAC)

- Polega za zastosowaniu tajnego klucza do stworzenia niewielkiego bloku danych o ustalonym rozmiarze, dołączanego do komunikatu.
- Kiedy A wysyła wiadomość do B, to oblicza MAC jako funkcję komunikatu i klucza. Wiadomość oraz MAC są przesyłane do odbiorcy.

MAC

Odbiorca również oblicza MAC. Jeżeli obie sumy są takie same i tylko obie strony znają tajny klucz, to:

- Odbiorca uzyskuje pewność, że wiadomość nie została zmodyfikowana.
- Odbiorca uzyskuje pewność, że wiadomość pochodzi od podanego nadawcy.
- Jeżeli komunikat zawiera numer kolejny (np. TCP, X.25), odbiorca uzyskuje pewność, że kolejność jest właściwa.

MAC

Jedną z najpopularniejszych sum kontrolnych jest algorytm oparty na DES, w którym dane podzielone na bloki 64-bitowe $D_1,\ldots,D_N$. Obliczamy:

$O_1=E_K(D_1)$,

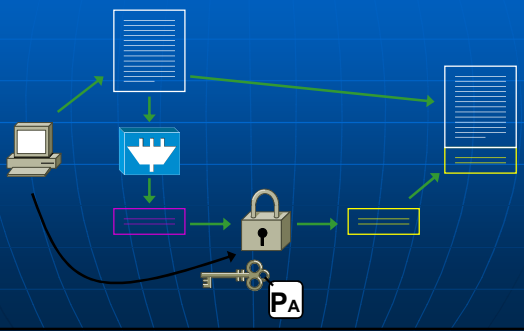
$O_2=E_K(D_2 \oplus O_1)$,

...,

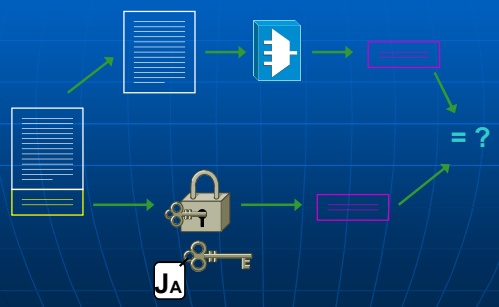
$O_N=E_K(D_N \oplus O_{N-1})$.

Ostatni wynik to suma MAC.

Funkcja skrótu a podpis cyfrowy



Weryfikacja podpisu cyfrowego



Algorytm hybrydowy

- Dane są szyfrowane algorytmem symetrycznym – kluczem sesji
- Dane są podpisywane – składany jest podpis cyfrowy (funkcja skrótu i algorytm asymetryczny)
- Klucz sesji jest często zmieniany
- Pozostaje problem bezpiecznego przesyłania klucza sesji –> kolejny wykład