Sieci TCP/IP - cz. 4

Zagadnienia bezpieczeństwa
Kryptograficzna ochrona aplikacji sieciowych

Ostatnia modyfikacja: 31.03.2020

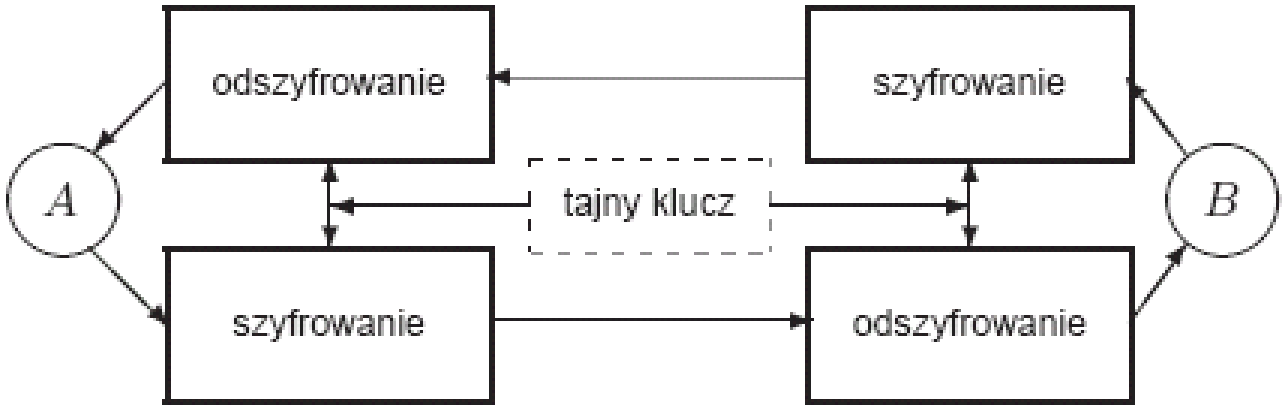
Wprowadzenie

- Znaczenie bezpieczeństwa komunikacji sieciowej rośnie: e-handel, przekazywanie decyzji, załatwianie spraw urzędowych przez sieć (np. składanie zleceń bankowych, zeznań podatkowych, zawieranie umów przez sieć).

- Podstawowe zadania kryptograficznej ochrony aplikacji sieciowych:
  - uwierzytelnianie nadawcy (*message origin authentication*) i odbiorcy
  - poufność danych (*data confidentiality*, *confidentiality of content*)
  - integralność danych (*data integrity*):
    - integralność zawartości wiadomości (*message content integrity*)
    - integralność sekwencji wiadomości (*message sequence integrity*)
  - niezaprzeczalność: nadania (*non-repudiation of message origin*) i odbioru wiadomości (*non-repudiation of message receipt*)
  - Kontrola dostępu (*access control*)

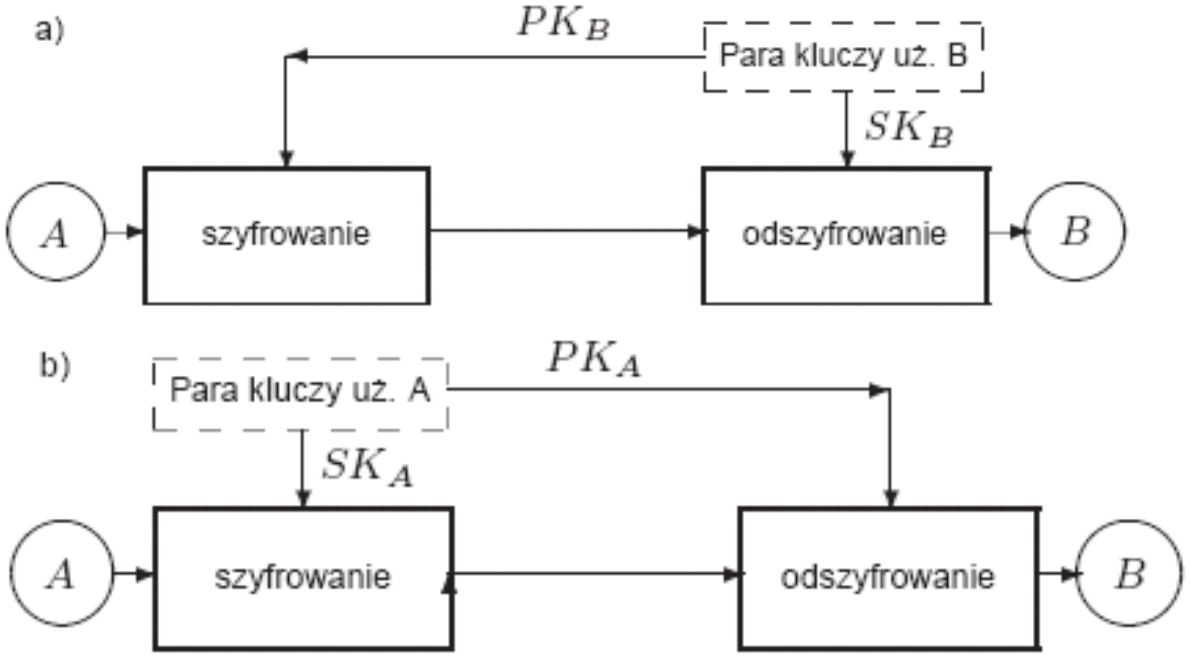
Elementarne techniki kryptograficzne

1. **Szyfrowanie**
   a) **Szyfrowanie symetryczne** (klasyczne), np.: DES, 3DES (TDES), IDEA, CAST, Blowfish, Twofish, SAFER, AES:
      - ten sam klucz służy do szyfrowania jak i do odszyfrowania wiadomości
      - dystrybucja klucza - przez bezpieczny kanał (np. z użyciem kuriera)
      - liczba kluczy dla `N` par użytkowników: `N*(N-1)/2`
      - względnie szybkie szyfrowanie/odszyfrowanie



Elementarne techniki kryptograficzne – c.d.

1. Szyfrowanie
   - **b. Szyfrowanie asymetryczne** (technika klucza publicznego), np. RSA, Diffie-Hellman, ElGamal, DSA/DSS
     - para kluczy: publiczny (jawny, PK) i prywatny (tajny, SK) wyznaczają wzajemnie odwrotne przekształcenia wiadomości
     - dwa tryby pracy: realizacja poufności odbioru (a), bądź niezaprzeczalności wysyłania wiadomości (b)



Elementarne techniki kryptograficzne – c.d.

2. Wyznaczanie kryptograficznego **skrótu wiadomości** (*message digest*, *message integrity code*), służy do kontroli integralności zawartości wiadomości i uwierzytelniania.

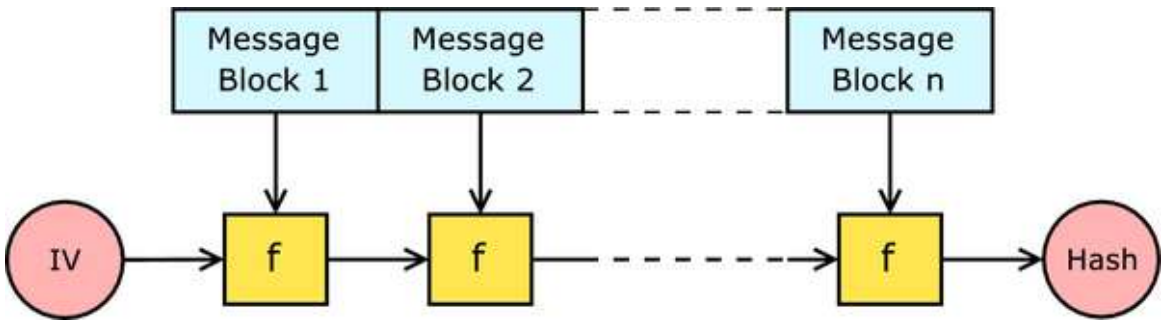
   Własności kryptograficznej funkcji skrótu:

   - funkcja skrótu nie zawiera tajnych parametrów i może być wyznaczona dla wiadomości `M` o dowolnej długości, `D=h(M)`
   - odtworzenie `M` na podstawie `D` jest niemożliwe obliczeniowo
   - znając funkcję `h` nie jest możliwe obliczeniowo wyznaczenie

     M_1, $M_2 \neq M_1$ takich, by $h(M_1) == h(M_2)$

   Skrót powinien mieć długość co najmniej 2 razy większą niż klucz algorytmu szyfrowania symetrycznego (dla danego zastosowania).

   Popularne algorytmy skrótu: SHA1 (160b – wychodzi z użycia), SHA2, SHA3 (224-512b), RIPEMD160, HAVAL (128-256b), MD5 (128b – nie jest polecany).



Wyznaczanie skrótu wiadomości o dowolnej długości wg Merkle-Damgarda

Elementarne techniki kryptograficzne – c.d.

3. **Generacja kryptograficznie bezpiecznych liczb losowych** - odgrywa dużą rolę w wielu procedurach (protokołach) kryptograficznych, np. utrudniając atak powtórzeniowy (patrz RFC4086 „Randomness Requirements for Security").

   Generatory programowe (liczb pseudo-losowych) mają skończony okres.

   Podstawowe wymagania na generatory kryptograficznie bezpieczne:

   - Nie istnieje algorytm o złożoności wielomianowej, który mógłby przewidzieć na podstawie znajomości `I` bitów generatora kolejny bit - z prawdopodobieństwem istotnie większym od 1/2.
   - Jeśli odgadnięty zostanie stan generatora w pewnym momencie, to niemożliwe jest odtworzenie wcześniej wygenerowanych bitów.

   Przykłady generatorów:

   - `/dev/random` - korzysta z losowości zdarzeń systemowych, może się blokować przy niedostatecznej entropii zarejestrowanych zdarzeń
   - `/dev/urandom` nigdy nie wyczerpujące się źródło (pseudo-)losowe.
   - Demon EGD (albo prng) – generatory pseudolosowe w przestrzeni użytkownika, wykorzystują losowość/nieprzewidywalność obciążonego systemu, a obserwowaną za pomocą programów takich jak 'w', 'last', 'vmstat'
   - Algorytm `Yarrow` (FreeBSD /dev/random) oraz algorytm `Fortuna` – wykorzystują algorytm szyfrowania symetrycznego do generacji ciągu liczb pseudolosowych, akumulatora zdarzeń systemowych, pliku zdarzeń losowych (dla szybkiego startu generatora po starcie systemu)
   - `Mersenne Twister` – bodaj najbardziej popularny algorytm generatora liczb pseudolosowych. Dla 32b implementacji okres: $2^{19937} - 1$

Realizacja usług ochrony informacji - poufność

- Szyfry:
  - Strumieniowe; jeśli `KS` – strumień klucza (*keystream*), a `M` – wiadomość (*message*), to kodowanie i dekodowanie opisują:

    `C[i]=KS[i]⊕ M[i]` oraz `M[i]=KS[i]⊕ C[i]`

    gdzie ⊕ to najczęściej `XOR`. Przykład szyfru strumieniowego: RC4
  - Blokowe (rozmiar bloku: 8-16 bajtów). Przykłady: AES, 3DES, RC2. Tryby pracy:
    - książki kodowej (ECB); wiadomość M jest dzielona na bloki `M[i]`; kodowanie: `C[i]=E(K,M[i]);` dekodowanie: `M[i]=D(K,C[i]),` gdzie `E` i `D` definiują operacje szyfrowanie i deszyfrowania, a `K` to klucz.

      Problem: takie same bloki wiadomości => takie same bloki szyfrogramu
    - z wiązaniem bloków (CBC):

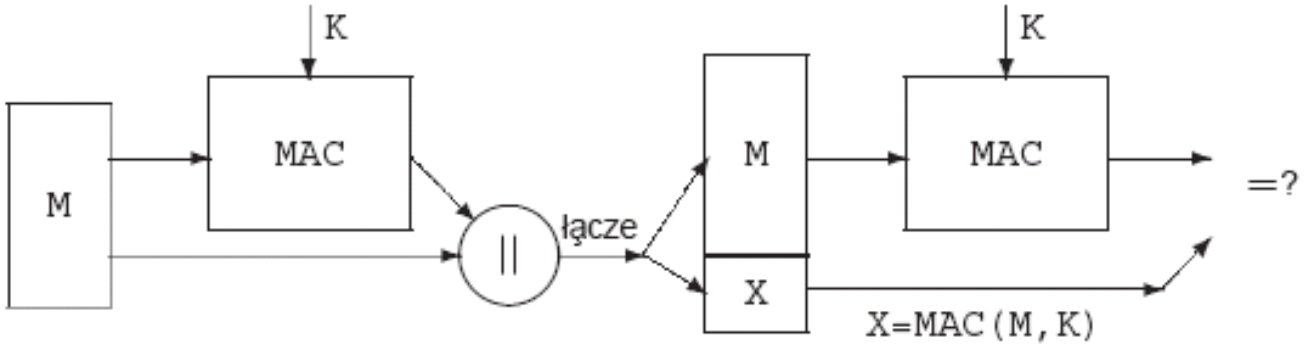
      `C[i]=E(K,M[i] ⊕ C[i-1])` gdzie `C[0]=IV` (random init. vector)

      `M[i]=D(K,C[i]) ⊕ C[i-1]`

      Uwaga: dla b-bitowych bloków wiadomość nie powinna mieć więcej niż `2^(b/2)` bloków (możliwe powtórzenie szyfrogramu, *rollover*)
    - ze sprzężeniem zwrotnym: z szyfrogramu (CFB), bądź z wyjścia (OFB)
- Zasada Kerckhoffsa: kryptosystem powinien opierać bezpieczeństwo nie tajności używanych w nich przekształceń danych (*security by obscurity*), ale na tajności klucza, wybierającego konkretne przekształcenie.
- Uważa się, że łatwiej o bezpieczną dystrybucję kluczy szyfrów asymetrycznych niż symetrycznych, ale generacja kluczy, szyfrowanie i deszyfracja są szybsze dla szyfrów symetrycznych. Przyjmuje się, że 1024b klucz algorytmu asymetrycznego daje siłę 80b klucza algorytmu symetrycznego.

Realizacja usług ochrony informacji - integralność

- Integralność pojedynczej wiadomości osiągana jest przez dołączanie znacznika integralności wiadomości (*Message Integrity Code*, *Modification Detection Code*), np. skrótu wiadomości.



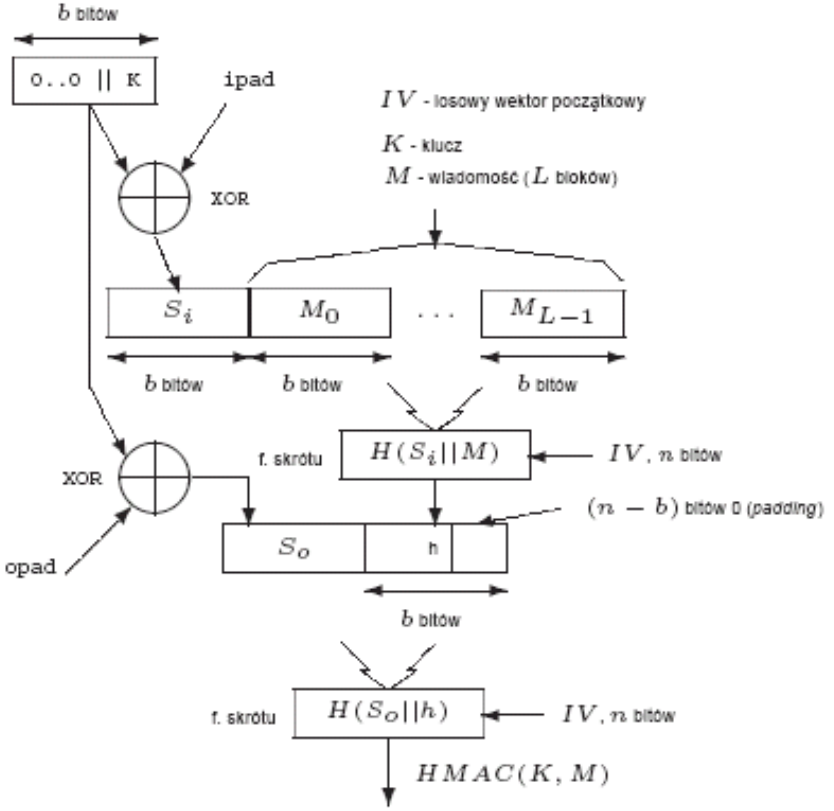
- Integralność sekwencji wiadomości jest chroniona przez:
  - zabezpieczanie każdej wiadomości przez znacznik integralności
  - używanie numerów sekwencyjnych wiadomości i ew. znaczników czasu nadania
  - potwierdzanie odbioru

Algorytm wyznaczania MAC - HMAC

- **HMAC** (*Hashed Message Authentication Code*, RFC2104, FIPS-198) z funkcją skrótu H

```
HMAC(K,M)=H((K⊕ opad)||
        H((K⊕ ipad)||M))
```

- `H(.)` – funkcja skrótu



- Operacje XOR ze stałymi `ipad` (`0x36` powtórzone `b/8` razy) i `opad` (`0x5c` powtórzone `b/8` razy) zamieniają miejscami (różne) połowy bitów argumentu. Typowo b=512.
- IPSec i TLS używają HMAC-SHA1 i HMAC-MD5

Realizacja usług ochrony informacji - uwierzytelnianie nadawcy

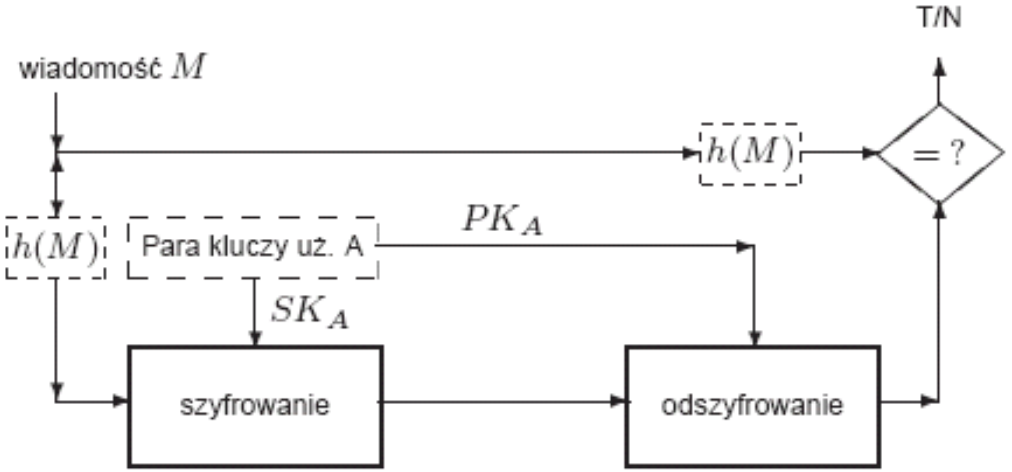
- Uwierzytelnienie nadawcy umożliwia sprawdzenie, czy nadawca jest tym użytkownikiem sieci, za którego się podaje.

- Uwierzytelnienie można zrealizować np. załączając do wiadomości znacznik uwierzytelnienia wiadomości (MAC, *message authentication code*), który zależy od wiadomości oraz **tajnego klucza** znanego tylko nadawcy i odbiorcy.

- Protokoły uwierzytelniania, nie korzystające z sekretów uprzednio znanych przez strony (np. tajnego hasła, certyfikatu), są narażone na atak ,,pośrednika" (*Man-in-the-middle*, MITM).

Realizacja usług ochrony informacji - niezaprzeczalność

- Niezaprzeczalność nadania wiadomości może być zrealizowana za pomocą podpisu cyfrowego, który jest funkcją wiadomości oraz informacji znanej tylko nadawcy.



Uwaga: dystrybucja kluczy publicznych musi gwarantować ważność klucza i jego pochodzenie.

Problem prostego modelu podpisu wiadomości: istnieje możliwość powtórnego użycia tego samego podpisu, np. do wielokrotnego dokonania tej samej transakcji (atak powtórzeniowy).

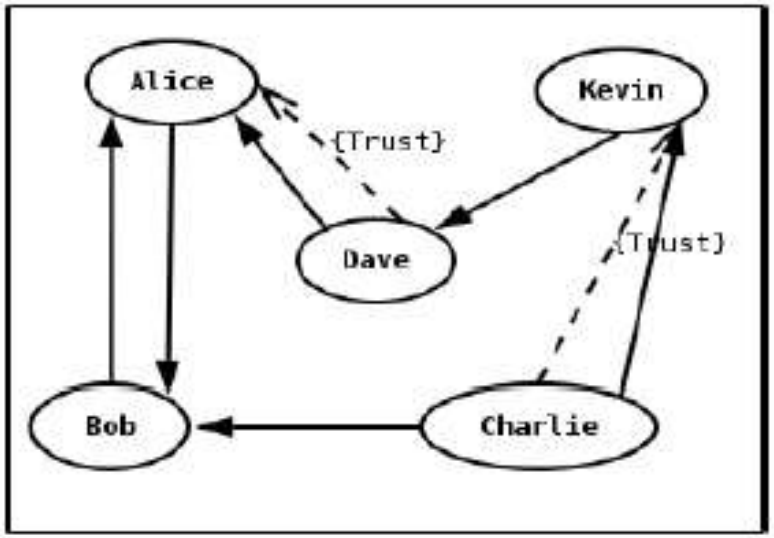
Rozwiązanie praktyczne: wiadomość powinna zawierać pole z numerem kolejnym wiadomości wysyłanej przez nadawcę (*sequence number*)

Realizacja usług ochrony informacji - uwagi

- Integralność, uwierzytelnienie i poufność – lepsze algorytmy symetryczne
- Niezaprzeczalność i wymiana kluczy - lepsze algorytmy niesymetryczne
- Ponieważ szyfrowanie asymetryczne jest nieefektywne, to zwykle wiadomości szyfruje się symetrycznie (np. algorytmem DES) używając wygenerowanego dla potrzeb sesji klucza, przesyłanego wraz z podpisem w postaci zaszyfrowanej (asymetrycznie) kluczem publicznym odbiorcy.

Uwierzytelnianie kluczy publicznych

1. **Rozproszona sieć zaufania** (*Web of trust model*) np. dla PGP:
   - każdy z użytkowników PGP ma swoją bazę kluczy publicznych (*public key ring*)
   - zapisana w bazie informacja o poziomie zaufania do klucza osoby ma wartości:
     - osoba nieznana (*unknown*)
     - osoba bez zaufania (*untrusted*)
     - osoba z minimalnym zaufaniem (*marginally trusted*)
     - osoba z pełnym zaufaniem (*completely trusted*)
   - klucz można wprowadzić ``ręcznie'' do bazy danych, deklarując zaufanie
   - nowy klucz może być wprowadzony ``automatycznie'' jeśli został podpisany przez dwie osoby o pełnym zaufaniu. (2 podpisy o minimalnym zaufaniu=1 podpis o pełnym zaufaniu), również z serwera kluczy (PGP >=5.0)



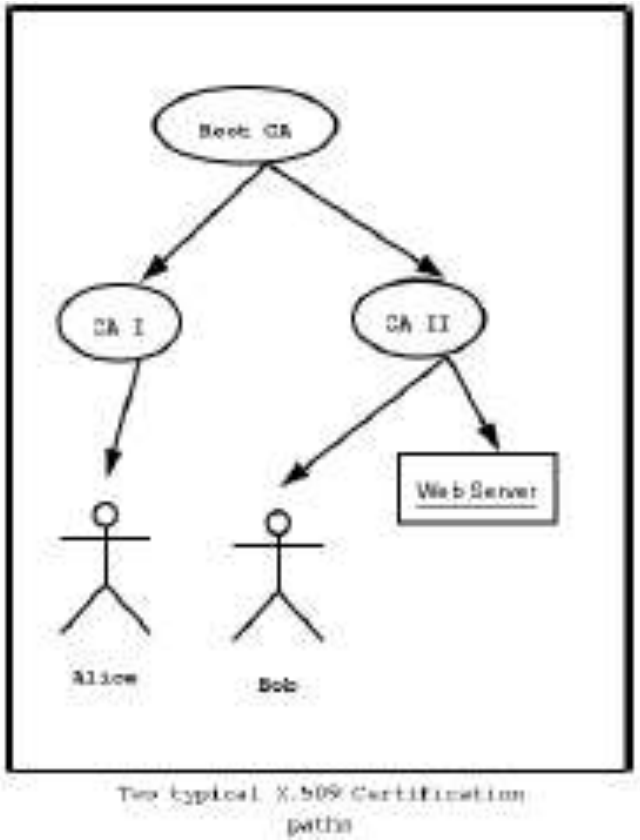
An example of the web of trust model

Rys. z: N. Mavrogiannopoulos, S. Josefssoon „GNU TLS”

Uwierzytelnianie kluczy publicznych

2. **Scentralizowany system uwierzytelniania** za pomocą certyfikatów, wydawanych przez zorganizowane hierarchicznie Urzędy ds Certyfikacji (Certification Authorities, CA), które:
   - rejestrują użytkowników, którzy chcą uzyskać świadectwa oryginalności swoich kluczy publicznych
   - odpowiadają na zapytania odnośnie prawdziwości związku danego klucza i użytkownika
   - prowadzą Listy Unieważnionych Certyfikatów (Certification Revocation List, CRL), aby umożliwić wycofywanie z użytkowania certyfikaty, których okres ważności nie upłynął.

   Każdy Urząd (CA) ma swój klucz prywatny (SK_{CA}) i publiczny (PK_{CA}) - znany wszystkim użytkownikom sieci. SK_{CA} służy do generowania certyfikatów, zawierających wiadomość łączącą (przez określony czas ważności) jawnie podaną tożsamość użytkownika i jego klucz publiczny, oraz podpis cyfrowy tej wiadomości (weryfikowalny z użyciem PK_{CA}).



Two typical X.509 Certification paths

Rys. z: N. Mavrogiannopoulos, S. Josefssoon „GNU TLS”

Standard ITU-T: **X.509** określa formaty certyfikatów list CRL oraz algorytm weryfikacji ścieżki uwierzytelniania certyfikatu.

Uwagi

- CA może zaświadczać o oryginalności certyfikatu innego CA; tworzą się łańcuchy certyfikacji (*Certificate chains*).

- Bezpieczeństwo przechowywania haseł tajnych (SK_{CA}) oraz dystrybucji haseł publicznych (PK_{CA}) urzędów na szczycie hierarchii, a posługujących się certyfikatami ``samopodpisanymi'' (*self-signed*), uwiarygodniają system certyfikacji.

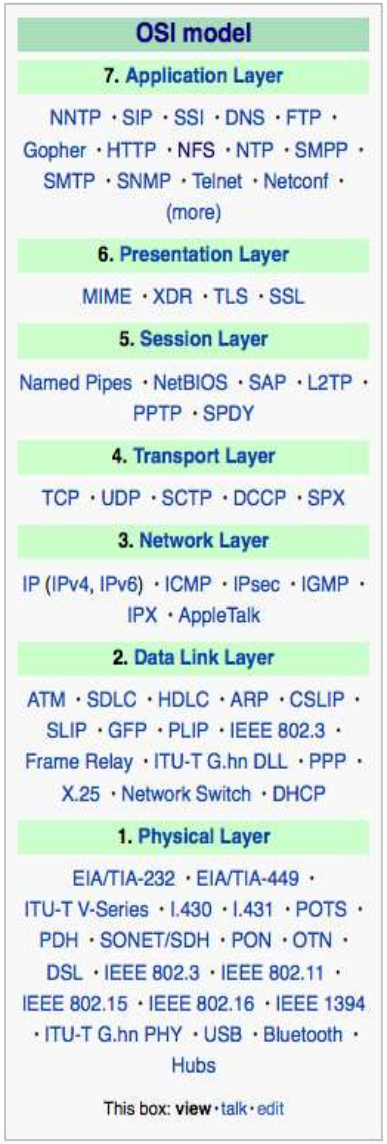
- Użytkownicy certyfikatów w momencie sprawdzania ważności podpisu Urzędu i okresu obowiązywania danego certyfikatu powinni sprawdzać również, czy certyfikat nie został umieszczony na Liście Unieważnionych Certyfikatów (CLR).

Kryptografia w protokołach sieciowych

Zabezpieczenia kryptograficzne występują w różnych warstwach modelu referencyjnego OSI/ISO

- W warstwie sieciowej typowo **IPSec + IKE** (wymiana kluczy).
  - oryginalny (zabezpieczany) pakiet IP jest szyfrowany, otrzymuje nowy nagłówek protokołu IPsec i w takiej formie jest przesyłany przez sieć.
  - Podstawa **Virtual Private Networks (VPNs)**

    Dwa kanały komunikacji:
    - kanał wymiany kluczy, za pośrednictwem którego przekazywane są dane związane z uwierzytelnianiem i szyfrowaniem (klucze)
    - kanał, który niesie pakiety transmitowane poprzez sieć prywatną.
- W warstwie prezentacji: **SSL/TLS.** Zapewnia poufność i integralność transmisji danych, a także uwierzytelnienie serwera, a niekiedy również klienta. Opiera się na szyfrowaniu asymetrycznym oraz certyfikatach X.509.



Grafika z: http://en.wikipedia.org/wiki/OSI_model

Bezpieczne tunelowanie z SSL

- Przykład (dla stunnel w wersji 3) `stunnel` – umożliwia utworzenie bezpiecznego tunelu komunikacji dla aplikacji używających TCP.
- `ssh` – umożliwia bezpieczną komunikację pomiędzy niezaufanymi hostami, ale również zdalne logowanie i zdalne wykonywanie poleceń.
- Wymagania programu `stunnel`:
  - Protokół połączeniowy (TCP nie UDP).
  - Protokół nie może używać wielu połączeń (jak np. FTP).
  - Protokół nie może wykorzystywać danych pozapasmowych (jak TELNET)
  - Odległy komputer nie może używać protokołu specyficznego dla aplikacji (jak SSLTELNET, gdzie SSL jest negocjowaną opcją).

```
rhost> ./tcpserv 7000
rhost> /usr/sbin/stunnel -f -d 7100 -r 7000 -p server.pem
localhost> ./s_tcpcli rhost 7100
```