Rozproszone i obiektowe systemy baz danych

Dr inż. Robert Wójcik

[lit] Mochnacki W., Kodowanie i kryptografia, Wrocław.

Wykład 9.

Bezpieczeństwo rozproszonych systemów baz danych

9.1. Systemy kryptograficzne

9.2. Zastosowania funkcji haszujących

9.3. Podpis cyfrowy – generowanie i weryfikacja

9.4. Szyfrowanie informacji w systemach rozproszonych

9.5. Metody uwierzytelniania podmiotów

9.6. System certyfikatów PKI

9.7. Bezpieczna komunikacja – protokół SSL

9.8. Bezpieczna komunikacja: protokół Needham-Schroeder symetryczny (uproszczony Kerberos).

System kryptograficzny:

- algorytm kryptograficzny,
- klucze kryptograficzne,
- metoda implementacji (sprzęt i oprogramowanie).

System kryptograficzny symetryczny (z kluczem tajnym)

W skład systemu kryptograficznego wchodzi pięć elementów:

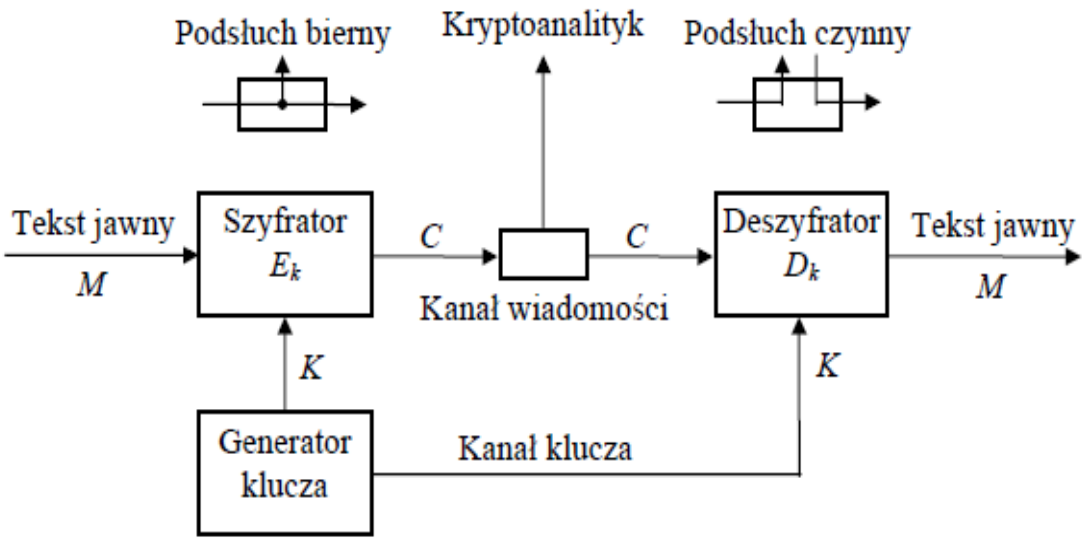
- przestrzeń wiadomości jawnych M,
- przestrzeń kryptogramów C,
- przestrzeń kluczy K,
- algorytm szyfrowania E,
- algorytm deszyfrowania D.

Elementy sprzętowe:

- szyfrator,
- deszyfrator,
- generator klucza.

Szyfrator realizuje algorytm kryptograficzny i generuje zbiór kryptogramów C z udziałem tajnego klucza kryptograficznego K (*przesyłany kanałem tajnym*).

Schemat blokowy systemu symetrycznego



W czasie transmisji kryptogramów może nastąpić ich przechwycenie przez kryptoanalityka.

Jeśli kryptoanalityk nie zmieni kryptogramu, to podsłuch taki nazywamy *biernym*, a jeśli nastąpi modyfikacja kryptogramu, podsłuch nazywamy *czynnym*.

W nowoczesnych systemach kryptograficznych algorytmy kryptograficzne nie są zazwyczaj tajne.

Tajne algorytmy okazały się niepraktyczne w dużych systemach i dlatego stosuje się je tylko w systemach o małym stopniu zabezpieczenia.

Przykładem stosowania tajnego algorytmu może być technika kodowania satelitarnych sygnałów telewizyjnych.

W powszechnych systemach kryptograficznych bezpieczeństwo systemu zapewnia tajny klucz.

Zastosowania kryptografii symetrycznej

- Szyfrowanie wiadomości i danych: $C = E_K(M)$

(dane na dyskach, wiadomości przesyłane w kanałach komunikacyjnych)

- Deszyfrowanie wiadomości: $M = D_K(C)$

Z definicji: $M = D_K(C) = D_K(E_K(M))$ – szyfrowanie jest przekształceniem odwracalnym.

Jednak bez znajomości klucza tajnego K odwrócenie przekształcenia jest niemożliwe w rozsądnym czasie.

- Szyfrowanie kluczem tajnym zabezpiecza wiadomości przed modyfikacjami (zapewnia integralność, autentyczność informacji).

- Szyfrowanie można traktować jak rodzaj kodowania: umożliwia wykrywanie błędów transmisji danych, gdyż przypadkowe modyfikacje kryptogramu uniemożliwią odszyfrowanie informacji.

- Uwierzytelnianie wiadomości: zaszyfrowanie wiadomości kluczem tajnym (prywatnym) jednoznacznie identyfikuje jej nadawcę (rodzaj podpisu).

- Uwierzytelnianie podmiotów: aby sprawdzić, czy dany podmiot jest tym za kogo się podaje należy zrealizować procedurę sprawdzającą, która polega na wysłaniu do podmiotu losowej liczby x, zaszyfrowanej algorytmem kryptografii symetrycznej oraz posiadanym kluczem tajnym K; badany podmiot jest tym za kogo się podaje, jeśli odeśle zaszyfrowaną wartość (x-1) lub zmodyfikowaną w inny sposób.

Wniosek:

W przypadku kryptografii symetrycznej poufność oraz integralność przesyłanych informacji jest zagwarantowana poprzez ich szyfrowanie oraz zapewnienie tajności klucza prywatnego.

System kryptograficzny asymetryczny (z kluczem publicznym)

Koncepcja systemu kryptograficznego z kluczem jawnym lub publicznym została przedstawiona pierwszy raz przez Diffiego i Helmana w 1976 r.

Algorytmy i klucze kryptograficzne:

- wiadomość jawna *M*,
- wiadomość zaszyfrowana (kryptogram) *C*,
- klucz ***K1*** (**klucz jawny**, publiczny wykorzystywany do szyfrowania wiadomości); *przesyłany kanałem otwartym*;
- klucz ***K2*** (**klucz tajny**, prywatny wykorzystywany do deszyfrowania wiadomości); *przesyłany kanałem tajnym*;
- algorytm szyfrowania *E*,
- algorytm deszyfrowania *D*.

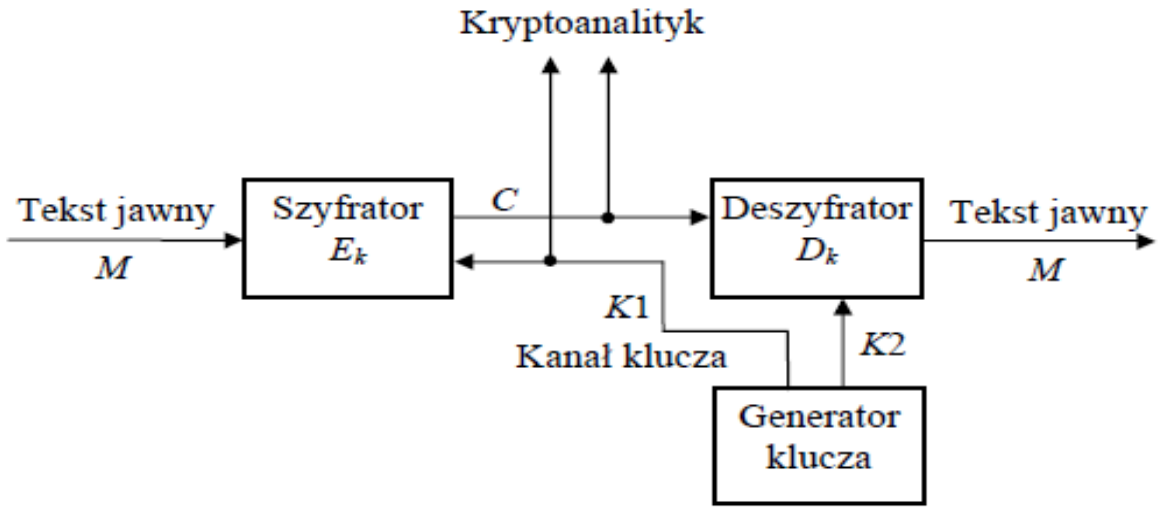
Elementy sprzętowe:

- szyfrator - realizuje algorytm szyfrowania *E*; generuje kryptogram C z wiadomości *M*, z udziałem klucza publicznego (jawnego) *K1;*

- deszyfrator – realizuje algorytm deszyfrowania *D*; odtwarza wiadomość jawną *M* z kryptogramu *C*, z udziałem klucza prywatnego (tajnego) *K2*;

- generator klucza: generuje klucze kryptograficzne.

Schemat blokowy systemu asymetrycznego



Zastosowania kryptografii asymetrycznej

- Szyfrowanie wiadomości i danych **kluczem publicznym K1**: **$C = E_{K1}(M)$**

(na ogół krótkie wiadomości ze względu na niską wydajność algorytmów kryptografii asymetrycznej oraz możliwy, znaczny wzrost rozmiaru wiadomości po zaszyfrowaniu).

- Deszyfrowanie wiadomości kluczem prywatnym: **$M = D_{K2}(C)$**

Z definicji: $M = D_{K2}(C) = D_{K2}(E_{K1}(M))$ – szyfrowanie jest przekształceniem odwracalnym. Jednak **bez znajomości klucza tajnego K2** odwrócenie przekształcenia jest niemożliwe w rozsądnym czasie.

Szyfrowanie kluczem tajnym K2 zabezpiecza wiadomości przed modyfikacjami (zapewnia integralność i autentyczność informacji);

- Uwierzytelnianie wiadomości: zaszyfrowanie wiadomości kluczem tajnym K2 jednoznacznie identyfikuje jej nadawcę (rodzaj podpisu).

$P = D_{K2}(M)$; podpis realizowany poprzez szyfrowanie całej wiadomości M kluczem prywatnym K2 nadawcy (w praktyce zastąpione podpisem cyfrowym, tj. szyfrowaniem haszu z wiadomości M);

$M = E_{K1}(P) = E_{K1}(D_{K2}(M))$; weryfikacja podpisu - źródła pochodzenia informacji, kluczem publicznym nadawcy wiadomości,
np. wysyłamy (M, $D_{K2}(M)$);

- Uwierzytelnianie podmiotów

Aby sprawdzić, czy dany podmiot jest tym za kogo się podaje należy zrealizować procedurę sprawdzającą, która polega na wysłaniu do podmiotu losowej liczby x, zaszyfrowanej kluczem publicznym K1 podmiotu; badany podmiot jest tym za kogo się podaje, jeśli odszyfruje wiadomość i odeśle wartość, np. (x-1) lub wartość (x-1) zaszyfrowaną naszym kluczem publicznym.

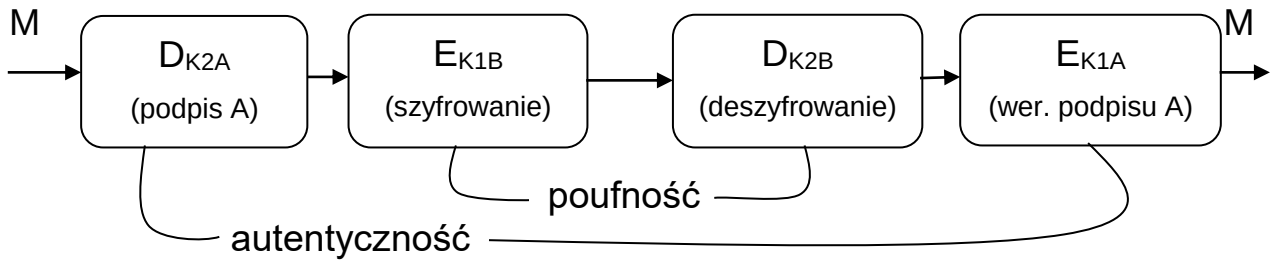
Wniosek:

W przypadku kryptografii asymetrycznej, aby zapewnić równocześnie poufność oraz integralność wiadomości przesyłanych od podmiotu A do podmiotu B, należy zastosować złożenie przekształceń szyfrowania i podpisywania z udziałem dwóch par kluczy (jawny, tajny): dla podmiotu A - (K1A, K2A) oraz dla podmiotu B – (K1B, K2B).

Wariant 1:

- podpis wiadomości M, kluczem prywatnym K2A podmiotu A (integralność), następnie szyfrowanie kluczem publicznym K1B podmiotu B (poufność);

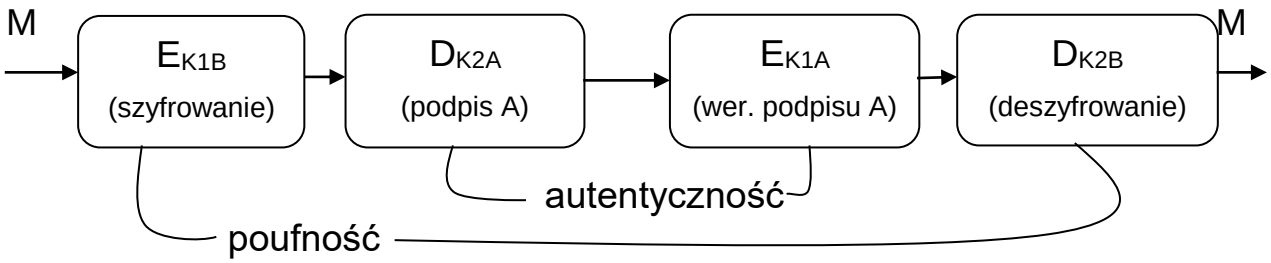
$P = D_{K2A}(M)$; $C = E_{K1B}(P)$; np. system ochrony poczty PGP;



Wariant 2:

- szyfrowanie wiadomości M, kluczem publicznym K1B podmiotu B (poufność), następnie podpis kluczem prywatnym K2A podmiotu A (integralność);

$C = E_{K1B}(M);$ $P = D_{K2A}(C);$ np. system ochrony poczty PEM;



Uwaga:

W obu przypadkach opcja szyfrowania całej wiadomości kluczem prywatnym nadawcy K2A może być zastąpiona podpisem cyfrowym (tj. hasz z wiadomości zaszyfrowany kluczem prywatnym nadawcy i dołączony do wiadomości).

Weryfikacja:

Wariant 1:

- odszyfrowanie wiadomości za pomocą klucza prywatnego K2B, następnie odczytanie wiadomości M kluczem publicznym K1A podmiotu A;

$$P = D_{K2B}(C); \quad M = E_{K1A}(P);$$

Wariant 2:

- odczytanie podpisu za pomocą klucza publicznego K1A, następnie odszyfrowanie wiadomości kluczem prywatnym K2B podmiotu B;

$$C = E_{K1A}(P); \quad M = D_{K2B}(C);$$

(szybszy do sprawdzenia; podpis nie pasuje to odrzucamy; mniej bezpieczny niż wariant 1).

Własności systemów kryptografii symetrycznej i asymetrycznej

Kryptografia symetryczna: szybsza w działaniu, bardziej odporna na złamanie; stosowna do szyfrowania strumieniowego oraz blokowego dużych ilości danych;

Kryptografia asymetryczna: wolna; może prowadzić do znacznego wzrostu rozmiaru danych po zaszyfrowaniu; stosowna do szyfrowania krótkich zbiorów danych, (np. wiadomości pocztowych, kluczy sesji dla kryptografii symetrycznej, sum kontrolnych (haszy) wiadomości.

Dyskusja ze studentami:

- rozpatrzyć wariant zabezpieczania wiadomości poczty elektronicznej poprzez ich szyfrowanie po stronie nadawcy za pomocą algorytmu kryptografii symetrycznej; tajny klucz sesji algorytmu symetrycznego jest generowany losowo po stronie nadawcy.

Zastosowania funkcji haszujących

W praktyce zapewnianie integralności informacji oraz uwierzytelnianie źródła ich pochodzenia poprzez szyfrowanie całych wiadomości za pomocą klucza prywatnego (tajnego) nadawcy jest nieefektywne dla dużych zbiorów danych.

W praktyce integralność wiadomości zapewnia się poprzez dodanie do wiadomości M jej haszu h(M) zaszyfrowanego kluczem prywatnym nadawcy, tj. zastosowanie **podpisu cyfrowego**.

Funkcja haszująca h umożliwia wyznaczenie dla danej wiadomości M ciągu bitowego, który stanowi rodzaj odcisku palca i w praktyce jest inny dla każdej wiadomości.

Funkcje haszujące są realizowane za pomocą tzw. funkcji jednokierunkowych, które posiadają następujące własności:

- dla każdej wiadomości M łatwo jest obliczyć h(M);
- h(M) ma zawsze stałą długość niezależnie od długości M;
- dla zadanego haszu X znalezienie M takiego, że h(M) = X, jest praktycznie niemożliwe;
- równie trudne jest znalezienie dla danej wiadomości M innej wiadomości M*, dla których funkcja haszu daje taki sam wynik, tj. h(M) = h(M*), gdzie $M \neq M^*$.

Podstawowe funkcje haszujące

Funkcja MD5: hasz128 bitów;

Funkcja SHA-1: hasz 160 bitów;

Funkcja SHA-2:
(inny algorytm niż SHA-1; warianty:
SHA-224, SHA-256, SHA-384, SHA-512 bitów);

Funkcja SHA-3: (warianty podobne do SHA-2, ale inny algorytm o lepszej wydajności obliczeniowej niż SHA-2).

Należy podkreślić, że w przypadku wymienionych funkcji może dochodzić do konfliktów, polegających na tym, że dwie różne wiadomości X i Y dadzą ten sam hasz h(X) = h(Y), ale jest to w praktyce zjawisko rzadkie.

Zastosowania funkcji haszujących

- podpis cyfrowy: hasz z wiadomości zaszyfrowany kluczem prywatnym nadawcy;
- zabezpieczanie umów przed zmianami (pozostawiamy hasz z odpowiednią datą u notariusza – **elektroniczny notariusz** lub ogłaszamy na stronie WWW);
- potwierdzenie istnienia dokumentu bez ujawniania jego treści (np. opis technologii, wzoru; dowód, że go posiadamy);
- zabezpieczanie przed modyfikacjami kodu programu przez wirusy (podczas pobierania programów sprawdzana suma kontrolna, np. md5).

Podpis cyfrowy – generowanie i weryfikacja

Podpis cyfrowy: hasz z wiadomości M zaszyfrowany kluczem prywatnym K2 nadawcy.

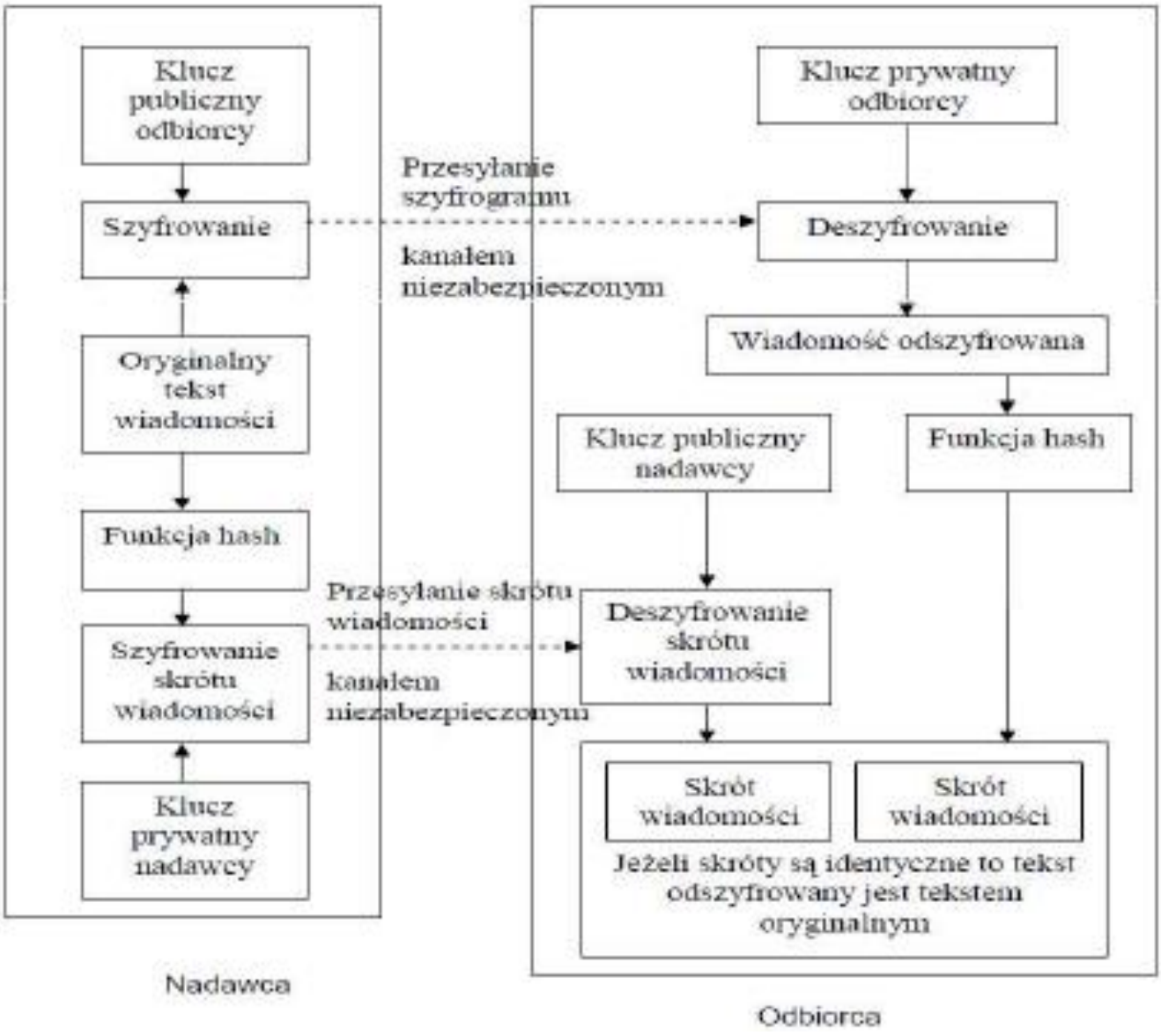
$P = D_{K2}(H(M))$; podpis realizowany poprzez szyfrowanie haszu H(M) z wiadomości M kluczem prywatnym K2 nadawcy.

Weryfikacja podpisu:

1) Wykonanie haszu (skrótu) z odebranej wiadomości.

2) Odszyfrowanie haszu (skrótu) zawartego w podpisie wiadomości kluczem publicznym nadawcy wiadomości.

3) Porównanie obu skrótów – powinny być identyczne.



Metody uwierzytelniania podmiotów

Podmiot – użytkownik, urządzenie, usługa, aplikacja, wymagająca bezpiecznej komunikacji.

Uwierzytelnianie (authentication) – proces sprawdzania, czy podmiot jest tym za kogo się podaje; inaczej weryfikacja tożsamości, poświadczanie tożsamości.

Podmioty – głównie urządzenia i użytkownicy systemów komputerowych (ludzie).

W aspekcie technicznym, problem uwierzytelniania dotyczy:

- uwierzytelniania maszyny przez maszynę

- uwierzytelniania człowieka przez maszynę.

Metody poświadczania tożsamości:

- słabe: wykorzystujące adres sieciowy lub hasło;

- silne: wykorzystujące dodatkowe urządzenia, mechanizmy i protokoły;

  - biometryczne (odcisk palca, rozkład naczyń,
    skan tęczówki oka);

  - oparte o tokeny, sms, hasła jednorazowe;

  - kryptograficzne:

    szyfrowanie liczb losowych z wykorzystaniem
    algorytmów symetrycznych lub asymetrycznych;
    rozbudowane w postaci protokołów uwierzytelniania
    typu „wyzwanie i odpowiedź” (challenge-response);

    zastosowanie certyfikatów kluczy publicznych;

Ochrona haseł

Uwierzytelnianie z wykorzystaniem haseł wymaga ich ochrony podczas przechowywania na komputerze lub przesyłania pomiędzy węzłami rozproszonej bazy danych.

Przechowywanie i przesyłanie hasła w postaci zakodowanej za pomocą:

- jednokierunkowej funkcji haszującej;

- uzupełnianie hasła liczbą losową znaną właścicielowi hasła (tzw. solą) i przechowywanie (przesyłanie) haszu ze zmodyfikowanego hasła;

- kodowanie hasła poprzez szyfrowanie algorytmem symetrycznym (np. DES, instrukcja crypt() Unix).

Protokoły uwierzytelniania typu wyzwanie i odpowiedź

Protokoły te często wykorzystują wartości jednorazowe *nonce*, które są wykorzystywane tylko raz w celu zapewnienia świeżości przesyłanych komunikatów i ochrony przed atakami podszywania się oraz ponownego użycia pakietów.

Rodzaje jednorazówek (nonce):

- Random number - liczba losowa; ma zagwarantować „świeżość” wiadomości;

- Sequence number - jednoznaczny, kolejny, niepowtarzalny numer wiadomości;

- Timestamp - znaczniki czasu wiadomości; pozwalają wykrywać opóźnienia i gwarantują właściwą kolejność wykonania operacji.

Protokół challenge-response (wyzwanie – odpowiedź)

Idea tego sposobu identyfikacji polega na odpowiedzi Alicji na wyzwanie przesłane przez Roberta, która przekona Roberta, że ma do czynienia rzeczywiście z Alicją.

Protokół challenge-response z kluczem tajnym

Protokół poświadczania tożsamości, wykorzystujący kryptografię symetryczną. W tym przypadku Alicja i Robert dysponują takim samym tajnym kluczem K (algorytm symetryczny).

1. Alicja komunikuje się z Robertem przedstawiając się jako Alicja.

2. Robert generuje liczbę losową r i wysyła ją Alicji.

3. Alicja szyfruje r za pomocą klucza K, powstałą wartość $E_K(r)$, przesyła do Roberta.

4. Jeśli Robertowi uda się poprawnie odszyfrować wiadomość od Alicji i otrzymana liczba będzie równa liczbie r, to Alicja jest tą osobą za którą się podaje.

W celu uzyskania dwustronnego uwierzytelnienia Robert i Alicja zamieniają się rolami.

Można również zrealizować wariant protokołu z użyciem funkcji haszującej H.

1. Alicja komunikuje się z Robertem przedstawiając się jako Alicja.

2. Robert generuje liczbę losową r i wysyła ją Alicji.

3. Alicja oblicza H(K, r) i powstałą wartość przesyła do Roberta.

4. Robert także oblicza H(K, r) i jeśli wynik zgadza się z wynikiem przysłanym przez Alicję to tożsamość Alicji zostaje potwierdzona.

Protokół challenge-response z kluczem publicznym

Protokół poświadczania tożsamości, wykorzystujący kryptografię asymetryczną. W tym przypadku Alicja dysponuje swoim kluczem prywatnym, a Robert posiada klucz publiczny Alicji.

1. Alicja komunikuje się z Robertem przedstawiając się jako Alicja.

2. Robert generuje liczbę losową r i wysyła ją Alicji.

3. Alicja szyfruje liczbę r swoim kluczem prywatnym i kryptogram wysyła do Roberta.

4. Robert deszyfruje kryptogram otrzymany od Alicji używając jej klucza publicznego i jeśli w wyniku otrzyma r, to tożsamość Alicji jest potwierdzona.

Szyfrowanie informacji w systemach rozproszonych

Do szyfrowania krótkich wiadomości można użyć algorytmów kryptografii asymetrycznej, np. RSA, ElGamala, np. wiadomości poczty elektronicznej PGP.

Zastosowanie algorytmów kryptografii asymetrycznej wiąże się z koniecznością potwierdzania źródła pochodzenia kluczy publicznych. Jednym z rozwiązań jest zastosowanie certyfikatów kluczy publicznych X.509 oraz infrastruktury klucza publicznego PKI wykorzystywanej do obsługi certyfikatów.

Do zabezpieczania informacji w bazach danych wykorzystuje się głównie algorytmy kryptografii symetrycznej: DES, 3DES, AES. Przed zapisem dane szyfruje się z użyciem klucza tajnego. W momencie odczytu są one deszyfrowane tym samym kluczem.

Np. W systemie baz danych MySQL 5.0 dostępne są następujące funkcje szyfrujące:

- AES_ENCRYPT(kolumna_lub_wyraz, klucz) - szyfruje słowo kluczem;
- AES_DECRYPT(kolumna_lub_wyraz, klucz) - odszyfrowuje słowo kluczem;

- ENCODE(kolumna_lub_wyraz, ’treść_hasła’) – szyfruje słowo hasłem:
- DECODE(kolumna_lub_wyraz, ’treść_hasła’) – odszyfrowuje słowo hasłem;

Certyfikat klucza publicznego wykorzystujący standard X.509

Podstawowy problem – upewnienie się, że klucz publiczny danego podmiotu rzeczywiście pochodzi od tego podmiotu.

Aby wyeliminować możliwość podstawienia fałszywego klucza publicznego danego podmiotu (podszycia się pod kogoś) zaprojektowano system certyfikatów klucza publicznego (PKI – Public Key Infrastructure).

Certyfikat jest to zbiór danych jednoznacznie identyfikujących dany podmiot (np. osobę, komputer) oraz pozwalający stwierdzić, czy ten podmiot, który legitymuje się tym certyfikatem jest rzeczywiście tym za kogo się podaje.

Certyfikaty podmiotów są potwierdzane (**podpisywane cyfrowo**) przez zaufane organizacje nazywane Urzędami Certyfikacji (Certyficate Authority – CA).

Przykładowe centra certyfikujące CA:

- Comodo CA,
- GeoTrust CA,
- Signet CA,
- Entrust CA,
- Thawte CA,
- VeriSign CA.

Certyfikat ma postać cyfrową – elektroniczną (certyfikat cyfrowy). Fizycznie jest to ciąg danych, zapisanych na odpowiednim nośniku (np. dysku, pamięci USB) w postaci pliku.

Certyfikat stanowi rodzaj elektronicznego zaświadczenia, które zawiera dane umożliwiające weryfikację przynależności klucza publicznego do danego podmiotu (np. certyfikat serwera banku, wykorzystywany do bezpiecznej komunikacji internetowej w oparciu o protokół SSL).

Źródło: www.signet.pl;

Jeden z głównych systemów certyfikacji opiera się o standard X.509. Certyfikat X.509 zawiera ciąg danych podpisanych cyfrowo przez określone CA, które wystawiło ten certyfikat. Aby zweryfikować autentyczność certyfikatu wystawionego przez dane CA, należy znać klucz publiczny tego CA (weryfikacja podpisu cyfrowego CA).

Klucz taki znajduje się na certyfikacie wystawionym dla CA przez jego nadrzędny organ certyfikujący, czyli nadrzędne CA. Zatem weryfikacja certyfikatu polega na prześledzeniu łańcucha zaufania od certyfikatu, poprzez pośrednie, nadrzędne centra CA, aż do głównego CA, cieszącego się powszechnym zaufaniem, które jako jedyne wystawia certyfikat sam dla siebie, tj. podpisuje go swoim kluczem prywatnym.

Łańcuch zaufania dla certyfikatu CERT:

CERT -> CA1 -> CA2 -> ... -> CAG_główne

Centra certyfikacji CA, wystawiające i sprzedające certyfikaty cyfrowe płacą za to, aby ich certyfikaty nadrzędne (root) zostały dodane do systemowej bazy certyfikatów w popularnych systemach operacyjnych (Windows, Linux, MacOS), przez co certyfikaty takie stają się zaufane dla użytkowników sieci Internet.

W praktyce przeglądarki internetowe rozpoznają i akceptują wystawiane przez te CA certyfikaty bez dodatkowych zapytań kierowanych do użytkowników (objawia się to zieloną kłódeczką w przeglądarce).
Najważniejsze informacje zawarte w certyfikacie X.509
(pola zapisane w postaci znaków ASCII):

1) Identyfikator wersji certyfikatu
2) Numer seryjny certyfikatu
3) Algorytm podpisu
4) Wystawca certyfikatu
5) Okres ważności od ... do
6) Nazwa właściciela certyfikatu (podmiot)
7) Klucz publiczny
8) Rozszerzenia: dodatkowe informacje, np. regulaminy certyfikacji, zasady użycia i zastosowanie klucza, punkty dystrybucji unieważnionych certyfikatów (CRL) i inne.
9) Podpis cyfrowy wystawcy certyfikatu

Certyfikaty pod Windows - uruchom: certmgr.msc .

Mogą być różne klasy certyfikatów w zależności od celu, do którego mają służyć (np. VeriSign):

Class 1 – dla podmiotów (osób) prywatnych, z przeznaczeniem dla poczty email.

Class 2 – dla organizacji, od których wymagane jest poświadczanie tożsamości.

Class 3 – dla serwerów i oprogramowania, zarządzającego certyfikatami kluczy publicznych wystawianymi przez CA.

Class 4 – dla transakcji biznesowych pomiędzy firmami.

Class 5 – dla zapewniania bezpieczeństwa organizacji prywatnych i rządowych.

Zastosowanie kryptografii klucza publicznego wymaga sprawnie działającej infrastruktury PKI.

Infrastruktura PKI służy do zarządzania cyfrowymi certyfikatami i kluczami szyfrującymi podmiotów (osób, programów i systemów).

Większość najważniejszych systemów bezpieczeństwa teleinformatycznego współpracuje z PKI. Są to m.in.: SSL (Secure Socket Layer), TLS , SMIME (Secure Multipurpose Internet Mail Extensions), SET (Secure Eletronic Transactions), IPSec (IP Security).

Infrastruktura PKI jest wykorzystywana w:

- bezpiecznej poczcie e-mail,
- transakcjach typu e-commerce (handel elektroniczny),
- wirtualnych sieciach prywatnych (Virtual Private Network - VPN),
- systemach ERP,
- zabezpieczeniach stacji roboczej użytkownika (m.in. zawartych na niej danych),
- zapewnieniu bezpieczeństwa witryn internetowych, urządzeń i aplikacji klienta.

Struktura PKI składa się z trzech głównych elementów:

- Urzędów Rejestracji (ang. Registration Authority - RA), dokonujących weryfikacji danych użytkownika a następnie jego rejestracji.

- Urzędów Certyfikacji (ang. Certification Authority - CA), wydających certyfikaty cyfrowe. Jest to poprzedzone procesem identyfikacji zgłaszającego się o wydanie certyfikatu. Pozytywne rozpatrzenie zgłoszenia kończy się wydaniem certyfikatu.

- Repozytoriów kluczy, certyfikatów i list unieważnionych certyfikatów (ang. Certificate Revocation Lists - CRLs). Typowa implementacja to umożliwienie, w oparciu o protokół LDAP, dostępu do certyfikatów i CRLs. Inne sposoby realizacji mogą być oparte na protokołach X.500, HTTP, FTP i poczcie elektronicznej.

Certyfikat może stać się nieważny przed datą jego wygaśnięcia, ze względu np. na zmianę nazwiska lub adresu poczty elektronicznej użytkownika, czy ujawnienie klucza prywatnego. W takich przypadkach CA odwołuje certyfikat i umieszcza jego numer seryjny na ogólnodostępnej liście CRL.

Struktura PKI jest tworzona w oparciu o **Główne Urzędy CA**.

Dla każdego z obszarów zastosowań (np. handel elektroniczny, sektor bankowo-finansowy, administracja publiczna), można tworzyć odrębne CA, podległe Głównemu Urzędowi.

Główne CA określa ogólną politykę certyfikacji, natomiast CA, obsługujące dany obszar zastosowań, odpowiada za jego politykę w tym zakresie. Może istnieć dowolna liczba CA, podległych głównemu Urzędowi CA, oraz dowolna liczba użytkowników.

Taka struktura tworzy hierarchię uwierzytelniania, która z kolei określa łańcuch certyfikatów, wiodący od użytkowników aż do cieszącego się ich zaufaniem Głównego CA.

Krajowa struktura PKI musi współdziałać ze strukturami PKI innych krajów, by zapewnić usługi o podobnym charakterze w kontaktach międzypaństwowych.

Podstawowe funkcje PKI

- *Rejestracja (ang. Registration)*

Użytkownik końcowy składa wniosek do Organu Rejestracji o wydanie certyfikatu. W tym celu dostarcza szereg informacji, wymaganych przez Kodeks Postępowania Certyfikacyjnego (Certification Practices Statement - CPS) danego CA. Dane te to np. nazwa własna podmiotu lub osoby wnioskującej o certyfikat, nazwa domenowa czy adres IP. Przed wystawieniem certyfikatu CA potwierdza (korzystając z wytycznych zapisanych w CPS) zgodność z prawdą danych, podanych przez użytkownika. Jeżeli o certyfikat ubiega się osoba fizyczna, CA weryfikuje także autentyczność własnoręcznego podpisu na wniosku o wydanie certyfikatu.

- *Certyfikacja (ang. Certification)*

Jeżeli dane, podane przez ubiegającego się o certyfikat, zostaną potwierdzone, CA wystawia nowy certyfikat (zawierający m.in. klucz publiczny posiadacza) i dostarcza go użytkownikowi. Jednocześnie klucz publiczny zostaje udostępniony wszystkim zainteresowanym poprzez złożenie go we właściwym repozytorium.

- *Generacja kluczy (ang. Key generation)*

Para kluczy (prywatny i publiczny) może zostać wygenerowana samodzielnie przez użytkownika końcowego, może on także powierzyć tę operację CA. W pierwszym przypadku użytkownik przesyła do CA jedynie swój klucz publiczny, w celu poddania go procesowi certyfikacji. Klucz prywatny pozostaje przez cały czas w rękach właściciela, dlatego też metodę tę uważa się za najbardziej bezpieczną. Jeżeli klucze generuje CA, są one dostarczane do użytkownika końcowego w sposób gwarantujący ich poufność, Wykorzystuje się do tego m.in. celu karty mikroprocesorowe (ang. smartcard), karty PCMCIA lub tokeny USB, zabezpieczone dodatkowym kodem PIN.

- *Odnawianie kluczy (ang. Key update)*

Wszystkie pary kluczy oraz skojarzone z nimi certyfikaty wymagają okresowego odnawiania. Jest to kolejne zabezpieczenie na wypadek ujawnienia klucza prywatnego skojarzonego z kluczem publicznym umieszczonym na certyfikacie. Wymiana kluczy jest konieczna, gdy:

- Upłynął okres ważności certyfikatu. Jest to sytuacja normalna, występująca regularnie co pewien czas (np. raz do roku). Odbywa się w możliwie krótkim czasie, bez dodatkowych formalności.

- Klucz prywatny, skojarzony z umieszczonym na certyfikacie kluczem publicznym, został skompromitowany. Jest to sytuacja wyjątkowa, a więc wymiana kluczy nie będzie już tak płynna i łatwa. W takich przypadkach CA odwołuje certyfikat poprzez umieszczenie jego numeru seryjnego na ogólnodostępnej liście CRL. Od tego momentu stary certyfikat traci ważność i rozpoczyna się procedura wystawiania nowego certyfikatu. Najgorszy przypadek dla każdego CA to kompromitacja klucza prywatnego jego Głównego CA (Root CA). W takim przypadku cała infrastruktura PKI podległa temu CA zostaje uznana za skompromitowaną i musi być tworzona od nowa.

- *Certyfikacja wzajemna (ang. Cross-certification)*

Ponieważ społeczność międzynarodowa nie stworzyła dotąd Globalnego Organu Certyfikacji (Global Root CA), powstało wiele Głównych Organów Certyfikacji (Root CA), początkowo nie powiązanych relacjami zaufania. Certyfikacja wzajemna rozwiązuje ten problem i pozwala użytkownikom z jednej struktury PKI ufać certyfikatom wystawianym przez CA z innej struktury. Główne CA z różnych struktur certyfikują się wzajemnie - może być to certyfikacja jednokierunkowa albo dwukierunkowa.

- *Odwołanie certyfikatu (ang. Revocation)*

Istnieją sytuacje, w których zachodzi potrzeba wcześniejszego odwołania certyfikatu. Powodem może być kompromitacja klucza prywatnego, zmiana nazwy przez użytkownika końcowego, bądź odejście pracownika z firmy, która wystawiła mu certyfikat. Zdefiniowana w standardzie X.509 metoda odwoływania certyfikatów wykorzystuje wspomniane już Listy Unieważnionych Certyfikatów (CRL), okresowo publikowane przez CA w repozytorium, w którym są przechowywane certyfikaty. Każdy certyfikat posiada swój unikalny numer seryjny, przypisany przez CA w momencie jego wystawiania. Lista CRL zawiera spis identyfikatorów odwołanych certyfikatów i jest opatrzona znacznikiem czasu, wystawionym przez CA.

- *Odzyskiwanie klucza (ang. Key recovery)*

Jest to dodatkowe zabezpieczenie na wypadek sytuacji, gdy użytkownik utraci swoje klucze. Jeżeli wszystkie klucze do szyfrowania albo negocjacji kluczy były przechowywane w bezpiecznym archiwum, będzie można je odzyskać i umożliwić dostęp do zaszyfrowanych danych. Najważniejsze jest zagwarantowanie, że klucze będzie mógł odzyskać tylko ich właściciel, nie zaś osoba trzecia.

Standaryzacja elementów PKI

Próbę zestandaryzowania funkcji PKI podjęła grupa robocza IETF (Internet Engineering Task Force), znana również jako grupa PKIX (PKI dla certyfikatów X.509).

Cztery podstawowe składniki modelu PKIX to:
- użytkownik,
- CA,
- RA,
- repozytorium certyfikatów.

Specyfikacja PKIX oparta jest na dwóch innych standardach:

- X.509 Międzynarodowego Związku Telekomunikacji (International Telecommunication Union - ITU) i

- PKCS Standardów Kryptografii Klucza Publicznego (Public Key Cryptography Standards) autorstwa RSA Data Security.

Najpopularniejsze standardy PKCS to:

- PKCS#7 - Cryptographic Message Syntax Standard (standard kryptograficznego kodowania wiadomości),
- PKCS#10 - Certificate Request Syntax Standard (standard kodowania wniosku o certyfikat),
- PKCS#12 - Personal Information Exchange Syntax Standard (standard kodowania informacji poufnych w postaci plików).

Ustanawianie klucza sesji z wykorzystaniem certyfikatu

Mechanizm ten stanowi podstawę działania protokołu SSL.

W tym przypadku klucz publiczny serwera WWW, uzyskany z jego certyfikatu, jest wykorzystywany do szyfrowania danych przesyłanych do serwera przez przeglądarkę internetową. Na podstawie danych wymienianych z serwerem w sposób bezpieczny (zaszyfrowany) ustalany jest tajny klucz sesji, który jest wykorzystywany do szyfrowania danych przesyłanych z serwera do przeglądarki.

Szczegóły tej koncepcji stanowią podstawę działania protokołu SSL/TLS.

Zasada działania protokołu SSL

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

SSL (Secure Socket Layer) jest protokołem szyfrowanej komunikacji stosowanym do realizacji bezpiecznych połączeń z serwerami sieciowymi.

Głównym zastosowaniem jest bezpieczna komunikacja przeglądarki internetowej z serwerem (HTTPS).

Protokół SSL został opracowany i zaimplementowany przez firmę Netscape, a jego dalszy rozwój odbywa się w ramach protokołu TLS (Transport Layer Security).

Protokół SSL składa się z czterech podprotokołów:

- protokół uzgadniania (SSL Handshake Protocol); odpowiedzialny za nawiązywanie połączenia;

- protokół zmiany specyfikacji szyfru (SSL Change Cipher); zmiana parametrów szyfru;

- protokół alarmowy (SSL Alert Protocol); obsługa błędów;

- protokół określania formatu pakietów (SSL Record Protocol); podział danych i tworzenie pakietów do wysłania.

Protokół SSL dodaje do protokołu sieciowego TCP/ IP warstwę ulokowaną pomiędzy warstwą transportową a warstwą aplikacji.

Warstwa aplikacji:
- protokół uzgadniania SSL;
- protokół zmiany specyfikacji szyfru SSL;
- protokół alarmowy SSL;
- http;
- inne protokoły (FTP, SMTP, POP3, IMAP4, telnet).

Protokół określania formatu pakietów

Warstwa transportowa
Warstwa sieci
Warstwa fizyczna
Protokół SSL może być wykorzystywany również w połączeniu z innymi protokołami niż http.

- HTTP (protokół HTTPS, port 443),
- FTP (sFTP, port 22),
- protokoły poczty elektronicznej,
  – Secure SMTP (SSMTP), port 465,
  – IMAP4 over SSL, port 993, port 585,
  – Secure POP3 (SSL-POP), port 995,
- telnet, protokół SSH, port 22.

Funkcje protokołu SSL:

• uwierzytelnienie stron biorących udział w transmisji,

• zapewnienie poufności transmisji poprzez szyfrowanie przesyłanych danych,

• zapewnienie integralności przesyłanych danych (zabezpieczenie danych przed ich modyfikowaniem w czasie transmisji).

Do uwierzytelnienia komunikujących się stron protokół SSL wykorzystuje infrastrukturę klucza publicznego (PKI, Public Key Infrastructure), standard X509.

Dla zapewnienia poufności transmisji wykorzystywane są:

- algorytmy asymetryczne RSA, Diffie-Hellmana, Fortezza's Key Exchange Algorithm, inne,
- algorytmy symetryczne DES, 3DES, RC4, inne,
- algorytmy haszujące MD5, SHA.

Za nawiązywanie połączenia w protokole SSL jest odpowiedzialny protokół uzgadniania.

Schemat działania protokołu wygląda następująco (jako **K** oznaczamy klienta, a jako **S** – serwer):

- **K → S ClientHello**
  Klient wysyła do serwera zgłoszenie zawierające m.in. obsługiwaną wersję protokołu SSL, dozwolone sposoby szyfrowania i kompresji danych oraz identyfikator sesji. Komunikat ten zawiera również liczbę losową używaną potem przy generowaniu kluczy.

- **K ← S ServerHello**
  Serwer odpowiada podobnym komunikatem, w którym zwraca klientowi wybrane parametry połączenia: wersję protokołu SSL, rodzaj szyfrowania i kompresji, oraz podobną liczbę losową.

- **K ← S Certificate**
  Serwer wysyła swój certyfikat pozwalając klientowi na sprawdzenie swojej tożsamości (ten etap jest opcjonalny, ale w większości przypadków występuje).

- **K ← S ServerKeyExchange**
  Serwer wysyła informację o swoim kluczu publicznym. Rodzaj i długość tego klucza jest określony przez typ algorytmu przesłany w poprzednim komunikacie.

- **K ← S ServerHelloDone**
  Serwer zawiadamia, że klient może przejść do następnej fazy zestawiania połączenia.

- **K → S ClientKeyExchange**
  Klient wysyła serwerowi wstępny klucz sesji, zaszyfrowany za pomocą klucza publicznego serwera. Na podstawie ustalonych w poprzednich komunikatach dwóch liczb losowych (klienta i serwera) oraz ustalonego przez klienta wstępnego klucza sesji obie strony generują klucz sesji używany do faktycznej wymiany danych. **Uwaga:** wygenerowany klucz jest kluczem algorytmu symetrycznego (np. 3DES, AES)! Jest on jednak ustalony w sposób bezpieczny i znany jest tylko komunikującym się stronom.

- **K → S ChangeCipherSpec**
  Klient zawiadamia, że serwer może przełączyć się na komunikację szyfrowaną.

- **K → S Finished**
  Klient zawiadamia, że jest gotowy do odbierania danych zaszyfrowanych.

- **K ← S ChangeCipherSpec**
  Serwer zawiadamia, że wykonał polecenie – od tej pory wysyłał będzie tylko zaszyfrowane informacje.

- **K ← S Finished**
  Klient od razu wypróbowuje mechanizm – ten komunikat jest już wysyłany bezpiecznym kanałem.

Uwierzytelnianie klienta

Jak widać na schemacie z poprzedniego punktu, w protokole SSL domyślna sytuacja zakłada tylko uwierzytelnianie serwera. Istnieją jednak metody pozwalające na uwierzytelnienie klienta. W tym celu korzysta się z trzech dodatkowych komunikatów:

- **K ← S CertificateRequest**
  Po przesłaniu swojego certyfikatu serwer zawiadamia, że chciałby otrzymać certyfikat klienta.

- **K → S Certificate**
  Po otrzymaniu komunikatu ServerHelloDone klient odsyła swój certyfikat.

- **K → S CertificateVerify**
  Klient musi potwierdzić jeszcze, że faktycznie posiada klucz prywatny odpowiadający wysłanemu certyfikatowi. W tym celu klient podpisuje swoim kluczem prywatnym skrót wszystkich dotychczas ustalonych danych o połączeniu i wysyła go korzystając z tego komunikatu.

Podczas wysyłania danych z użyciem protokołu SSL wykonywane są operacje:

- dane dzielone są na pakiety;

- pakiety poddawane są kompresji;

- dla każdego pakietu obliczany jest kod MAC;

- dane są łączone z ich kodem uwierzytelniającym MAC;

- uzyskane pakiety są szyfrowane;
- pakiety są łączone w pakiety TCP i wysyłane do sieci.

Za przeprowadzenie tych operacji jest odpowiedzialny protokół określania formatu pakietów. Jeśli w trakcie wykonywania tych operacji pojawi się pakiet z błędami, to uruchamiany jest protokół alarmów. W przypadku, gdy pojawi się informacja typu ostrzeżenie lub błąd na połączenie nakładane są ograniczenia, w przypadku błędu krytycznego połączenie jest przerywane.

Algorytmy szyfrowania pakietów są ustalane w fazie nawiązywania połączenia. Komputer klienta podaje listę obsługiwanych metod, spośród których serwer wybiera najbezpieczniejszą dla danego połączenia. Do szyfrowania danych wykorzystywane są algorytmy kryptograficzne symetryczne (np. AES), natomiast do przesyłania kluczy sesji stosowane są algorytmy asymetryczne.

Protokół Needham-Schroeder symetryczny

Rozważamy sieć złożoną z n węzłów, w których znajdują się podmioty, które chcą się bezpiecznie komunikować, wykorzystując kryptografię symetryczną.

Podmioty (i, j), które chcą się bezpiecznie komunikować wymagają jednego klucza tajnego k_{ij}. W przypadku n podmiotów niezbędny jest jeden klucz dla każdej pary podmiotów:

$$\binom{n}{2} = n * (n-1)/2 \ .$$

Np. dla trzech podmiotów 1, 2, 3 potrzebne są 3 klucze: k12, k13 oraz k23.

1 ---- k12 ----- 2 ----- k23 ------ 3 ----- k13 ----- 1

Liczba kluczy rośnie z kwadratem liczby węzłów.

Rozwiązaniem problemu zarządzania kluczami w środowisku rozproszonym złożonym z n podmiotów jest centrum dystrybucji kluczy KDC.

Centrum KDC posiada klucze symetryczne do komunikacji z każdym z podmiotów w1, w2, …, wn.

Aby zapewnić bezpieczną komunikację pomiędzy parą podmiotów (wi, wj) centrum KDC generuje symetryczny klucz sesji s_{ij}, wykorzystywany do szyfrowania połączeń pomiędzy podmiotami, który jest dostarczany do podmiotów w sposób bezpieczny z wykorzystaniem protokołu Needhama-Schroedera.

Protokół ten zapewnia bezpieczną komunikację oraz wzajemne uwierzytelnienie podmiotów. Wykorzystuje on pojęcie jednorazówki. Jest to wartość, która w protokole może być wykorzystana tylko raz.

Niech A będzie podmiotem, który chce się bezpiecznie komunikować z usługą B, np. serwerem bazodanowym, usługą pocztową. Podmiot A posiada tajny, symetryczny klucz K_{AC}, służący do komunikacji z KDC, natomiast podmiot B posiada klucz K_{BC}. Klucze K_{AC} i K_{BC} znajdują się również w KDC.

Celem protokołu jest obopólne poświadczenie tożsamości pomiędzy usługą B, a podmiotem A z wykorzystaniem centrum KDC, a także bezpieczna komunikacja A z B.

Niech N_A , N_{A1} oraz N_B będą jednorazówkami generowanymi odpowiednio przez A i B.

K_{AB} jest kluczem symetrycznym, generowanym przez KDC i służącym jako klucz sesji do komunikacji A z B.

Schemat wymiany komunikatów pomiędzy KDC i podmiotami jest następujący.

1) Podmiot A wysyła do KDC komunikat, w którym identyfikuje siebie oraz usługę B, z którą chce się komunikować; dodatkowo umieszcza w komunikacie jednorazówkę N_A, zapewniającą świeżość komunikatu.

$A \rightarrow KDC: A, B, N_A;$

2) W odpowiedzi serwer KDC odsyła komunikat zaszyfrowany kluczem K_{AC}, w którym umieszcza klucz K_{AB}, jednorazówkę N_A, identyfikator B, oraz bilet dostępu do B zaszyfrowany kluczem usługi K_{BC}.

$KDC \rightarrow A: \{ N_A, K_{AB}, B, \{ K_{AB}, A \}_{KBC} \}_{KAC}$

3) Podmiot A odszyfrowuje komunikat swoim kluczem K_{AC} i uzyskuje klucz K_{AB} oraz bilet dostępu do usługi B. Podmiot A wysyła do B bilet oraz jednorazówkę N_{A1} zaszyfrowaną kluczem sesji.

$A \rightarrow B: \{ K_{AB}, A \}_{KBC} \ ; \ \{ N_{A1} \}_{KAB}$

4) Usługa B odszyfrowuje bilet i uzyskuje klucz sesji K_{AB}, który wykorzysta do bezpiecznej komunikacji z A. Usługa B uwierzytelnia się przesyłając zmodyfikowaną jednorazówkę.

$B \rightarrow A: \{ N_{A1} - 1 \}_{KAB}$

Aby zapewnić obopólne uwierzytelnienie usługa B wysyła do A jednorazówkę N_B zaszyfrowaną kluczem sesji.

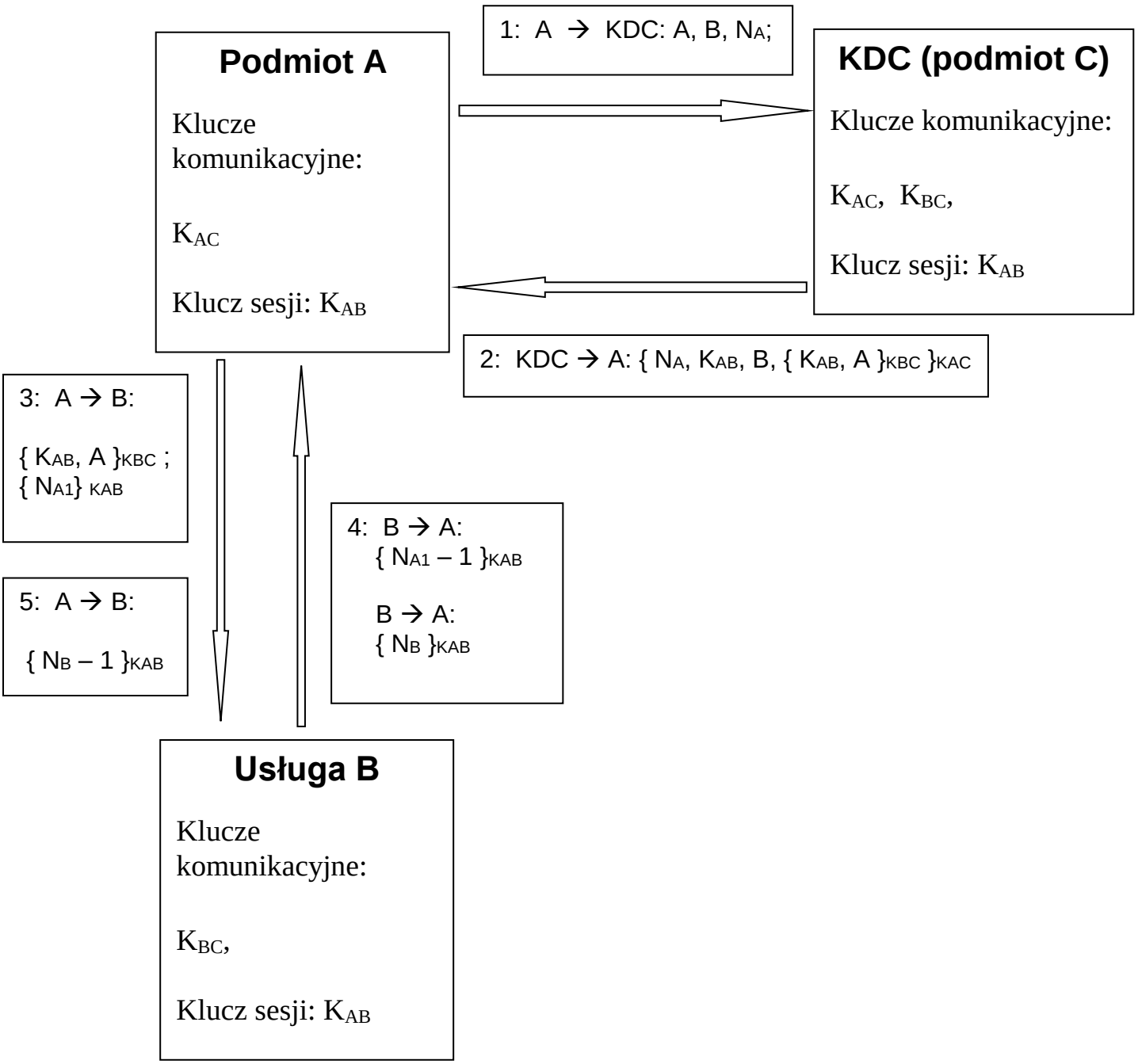
$B \rightarrow A: \{ N_B \}_{KAB}$

5) Podmiot A uwierzytelnia się odpowiadając

$A \rightarrow B: \{ N_B - 1 \}_{KAB}$

Od tej pory możliwa jest bezpieczna komunikacja pomiędzy A i B z wykorzystaniem symetrycznego klucza sesji K_{AB} (który jednak może zostać złamany).

Protokół N-Sch



Gdyby klucz sesji K_{AB} został skompromitowany możliwe byłoby podszycie się pod A poprzez ponowne przesłanie biletu $\{ K_{AB}, A \}_{KBC}$ do B, który nie byłby w stanie wykryć, że jest to bilet nieaktualny. Aby temu zapobiec można dodać jednorazówkę N_{B1} do biletu przesyłanego do B.

1) Najpierw A wysyła do B żądanie nawiązania komunikacji:

$A \rightarrow B: A$

2) B odpowiada jednorazówką N_{B1} zaszyfrowaną kluczem przeznaczonym do komunikacji z serwerem KDC:

$B \rightarrow A: \{ A, N_{B1} \}_{KBC}$

3) Podmiot A wysyła do KDC komunikat, w którym identyfikuje siebie oraz usługę B, z którą chce się komunikować; dodatkowo umieszcza w komunikacie jednorazówkę N_A, zapewniającą świeżość komunikatu oraz jednorazówkę od B:

$A \rightarrow KDC: A, B, N_A, \{ A, N_{B1} \}_{KBC}$

4) W odpowiedzi serwer KDC odsyła komunikat zaszyfrowany kluczem K_{AC}, w którym umieszcza klucz K_{AB}, jednorazówkę N_A, identyfikator B, oraz bilet dostępu do B zaszyfrowany kluczem usługi K_{BC} oraz uzupełniony o jednorazówkę N_{B1}.

$KDC \rightarrow A: \{ N_A, K_{AB}, B, \{ K_{AB}, A, N_{B1} \}_{KBC} \}_{KAC}$

5) Podmiot A odsyła bilet do B. Jednorazówka N_{B1} zapewnia świeżość biletu i zapewnia B, że jest to bilet dla A wygenerowany w odpowiedzi na aktualne żądanie.

$A \rightarrow B: \{ K_{AB}, A, N_{B1} \}_{KBC} \; ; \; \{ N_{A1} \}_{KAB}$

Powtórzenie biletu wymagałoby przesłania wiadomości postaci $\{ K_{AB}, A, N_{B1} \}_{KBC}$ z nową, świeżą jednorazówką N_{B1}, czego bez znajomości klucza K_{BC} nie da się zrobić. Powtórzenie jednorazówki N_{B1} oznaczałoby, że bilet jest nieaktualny.