Technologie zapewniające bezpieczeństwo w systemach operacyjnych (PKI, Smart Cards, SSL, SSH)

Łukasz Dudek
Paweł Koziński

Systemy operacyjne 2002/2003

15 stycznia 2003

Spis treści

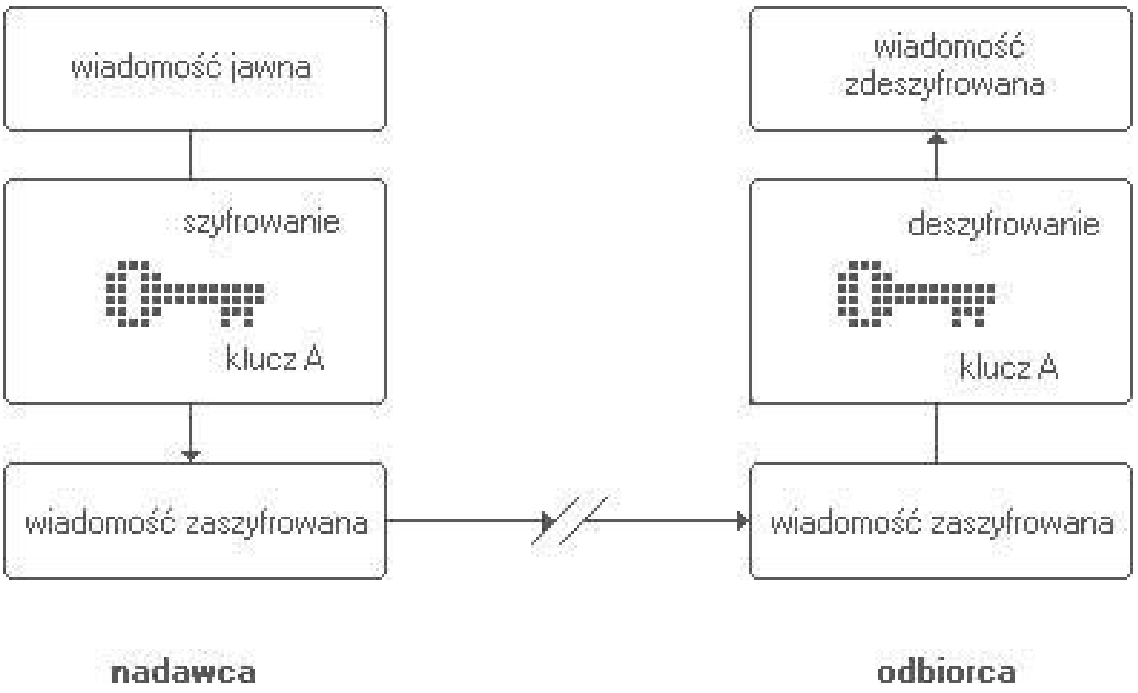
1 PKI

1.1 Wprowadzenie

Elementem PKI czyli Infrastruktury Klucza Publicznego (ang. Public Key Infrastructure), o którym mówi się ostatnio najgłośniej jest podpis elektroniczny. 16 sierpnia 2002 roku weszła w życie ustawa o podpisie elektronicznym (po tym jak wcześniej została zaskarżona do Trybunału Konstytucyjnego) sprawiając tym samy, że wiele potencjalnych zastosowań Internetu może zostać wykorzystanych w życiu codziennym. Idea podpisu elektronicznego bazuje na zaawansowanych technologiach kryptograficznych wykorzystujących klucze publiczne.

1.2 Podstawy kryptograficzne

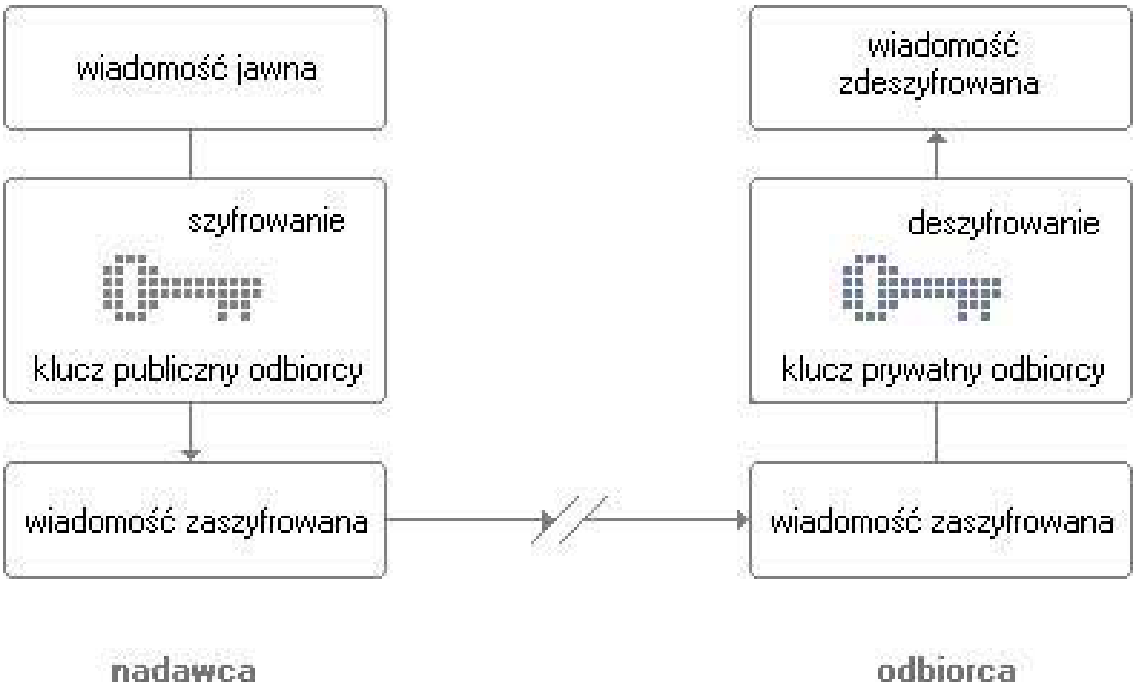
Poczatkowo metody szyfrowania opierały sie na algorytmach wykorzystujących tylko jeden klucz kryptograficzny zarówno do zaszyfrowania jak i odszyfrowania wiadomości. Algorytmy takie są szybkie, często wspierane sprzętowo. Największym problemem związanym ze stosowaniem szyfrowania z kluczem symetrycznym jest konieczność istnienia bezpiecznego kanału umożliwiającego niezagrożoną wymianę klucza. Większość ataków na systemy bazujące na technologiach kryptograficznych z kluczem publicznym są to ataki na kanał którym przesyłany jest klucz lub też ataki "brute force". Algorytmy szyfujące oparte na kluczu symetrycznym to: DES, DES, Blowfish, IDEA czy najmłodszy AES.



Rys. 1 Klucz symetryczny

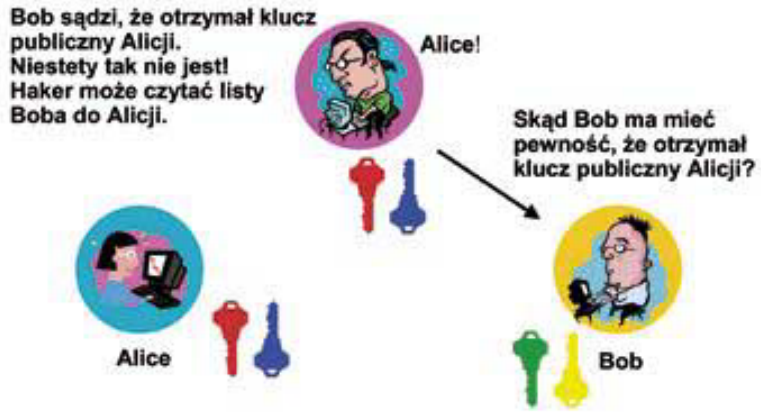
Wprowadzenie do kryptografii klucza asymetrycznego możliwe było dopiero po rewolucji jaką było opracowanie algorytmu RSA (choć sama idea klucza asymetrycznego znana była dużo wcześniej). W algorytmach z kluczem asymetrycznym wykorzystywane sa różne klucze do szy-

frowania i odszyfrowania informacji. Do szyfrowania wiadomości wykorzystywany jest klucz publiczny odbiorcy zaś do odszyfrowania klucz prywatny odbiorcy.



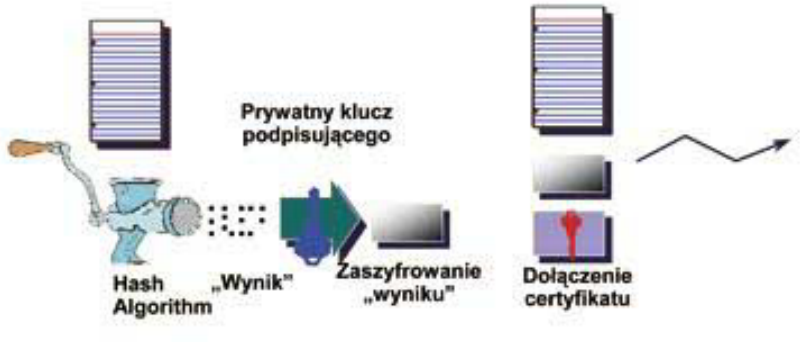
Rys. 2 Klucz asymetryczny

Zagrożeniem dla technologii klucza asymetrycznego jest tzw. atak "man in the middle". Polega on na wcięciu sie przez atakującego do kanułu, którym przesyłane są informacje i przesłanie stronom swojego klucza publicznego tak jakby byl to klucz drugiej strony. W ten sposób atakujący może swoim kluczem prywatnym odszyfrować przesyłane wiadomości i ponownie ja zaszyfrować. Aby zapobiec takiej sytuacji należy zapewnić kanał weryfikacji prawdziwości kluczy kryptograficznych.



Rys. 3 Atak "Man in the middle"

1.3 Podpis cyfrowy



Rys. 4 Podpis cyfrowy

Na opisanych technikach kryptograficznych opiera się idea podpisu elektronicznego. Podpisanie dokumentu odbywa się w nastepujących krokach

1. Wyznaczenie wartości funkcji haszującej dla zadanego dokumentu.
2. Zaszyfrowanie otrzymanej wartości kluczem prywatnym.
3. Scalenie tak uzyskanej sygnatury z resztą dokumentu.

Weryfikacja prawdziwości podpisu przeprowadzana jest następująco:

1. Sygnatura zostaje oddzielona od dokumentu.
2. Dla dokumentu obliczana jest wartość funkcji haszującej.
3. Sygnatura jest odszyfrowywana przy użyciu klucza publicznego.
4. Jeśli obie wartości są równe to podpis jest prawdziwy.

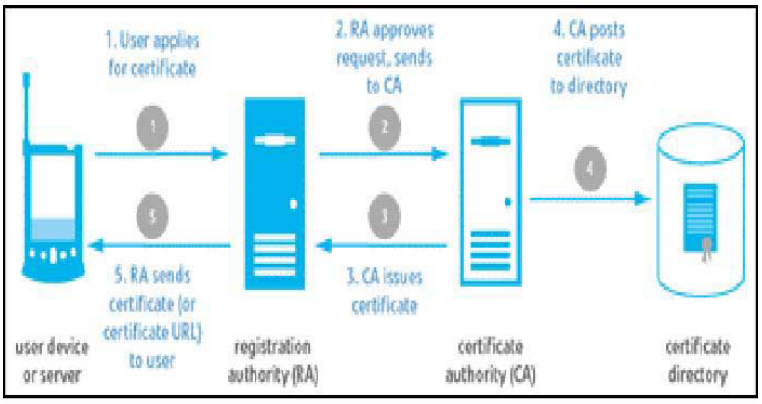
1.4 Infrastruktura klucza publicznego

Dla prawidłowego funkcjonowania PKI niezbędne są certyfikaty cyfrowe potwierdzające prawdziwość kluczy kryptograficznych uczestników transakcji. Prawdziwość certyfikatów gwarantowana jest przez instytucje certyfikujące, co do których podmioty mają zaufanie (np za sprawą odpowiedniej ochrony czy regularnego audytowania).

Instytucje stanowiące trzon PKI to:

1. Urzędy Rejestracji (ang. Registration Authority - RA), odpowiedzialne są za rejestrację użytkowników
2. Urzędów Certyfikacji (ang. Certification Authority - CA), odpowiedzialnę są za wydawanie certyfikatów cyfrowych (po wcześniejszej identyfikacji podmiotu występującego o wydanie takiego certyfikatu)

3. Repozytoria kluczy, certyfikatów i list unieważnionych certyfikatów (ang. Certificate Revocation Lists - CRLs).



Rys. 5 Wydawanie ceryfikatów
PKI zapewnia następujące usługi uwierzytelniania

- uwierzytelnianie podmiotów bedących stronami transakcji
- uwierzytelnianie danych
- integralność danych (łatwo określić czy podpisana informacja została zmieniona)
- niezaprzeczalność (podmiot nie może wyprzeć się faktu nadania wiadomości)

1.5 Funkcje PKI

Podstawowe funkcje PKI to:

1. Rejestracja (ang. Registration)
   Użytkownik zgłasza się sie do urzędu rejestracji, CA weryfikuje jego dane

2. Certyfikacja (ang Certification)
   W przypadku pomyślnej weryfikacji danych użytkownikowi zostaje wydany certyfikat cyfrowy, certyfikat zostaje złożony w repozytorium publicznym tak by był dostępny dla wszystkich zainteresowanych

3. Generacja kluczy (ang. Key generation)
   Klucze mogą zostać wygenerowne przez użytkownika jak i CA (w tym przypadku częto dostarczane są użytkownikowi przez wykorzystanie Smart Cards)

4. Odnawiania kluczy (ang. Key update)
   Klucze wymagają odnowienia po upływie terminu ważności lub w momencie gdy ujawniony został klucz prywatny skojarzony z danym kluczem publicznym

5. Certyfikacja wzajemna (ang. Cross-cerification)
   Pozwala użytkownikom z jednej struktury PKI ufać użytkownikom drugiej struktury
6. Odwołanie certyfikatu (ang. Revocation)
7. Odzyskiwanie klucza (ang. Key recovery)

   Zabezpieczenie na wypadek gdy użytkownik utraci swoje klucze

2 Smart Cards

* plastikowa karta podobna do karty kredytowej posiadająca wbudowany mikoroprocesor oraz pamieć
* Smart Cards znajdują szerokie zastosowanie w różnych technologiach zapewniających bezpieczeństwo
* dane przechowywane na Smart Cards mogą być bardziej złożone niż w przypadku kart z paskiem magnetycznym, mogą być też modyfikowane
* dane na Smart Cards są zapisywane do drzewiastej struktury katalogów
* mają ograniczony kontakt ze światem zewnętrznym
* są często wykorzystywane do przechowywania takich danych jak klucze prywatne
* dodatkowo moga być zabezpieczone przez PIN (i nie tylko)
* występują Smart Cards kontaktowe i bezkontaktowe (połączenie poprzez fale radiowe)
* Smart Cards moga być wykorzystywane przy uwierzytelnianiu w SSH
* Smrt Cards znajdują zastosowanie w połączeniu z Kerberosem
* trzy platformy dominują na rynku oprogramowana dla Smart Cards
    - MULTOS (wykorzystywany gdy bezpieczeństwo odgrywa duże znaczenie)
    - Java (JavaCard Operating System)
    - Microsoft (Microsoft Windows for Smart Cards)
* Smart Cards moga znaleźć zastosowanie jako klucz do drzwi, portfel elektroniczny, karta biblioteczna itp. wszystkie te funkcje może spełnić jedna karta)
* problemem dotyczących Smart Cards jest ciągły brak pewnych standardów, które zapewnią większą przenośność oraz szersze zastosowanie

3 SSL

3.1 Wprowadzenie

SSL (Security Socket Layer) to protokół bezpiecznej komunikacji zaprojektowny przez firmę Netscape. Do szyfrowania połaczeń wykorzystuje technologię kluczy publicznych i prywatnych.

Zadania realizowane przez SSL to zapewnienie:

- autoryzacji (poprzez określenie tożsamości zarówno serwera jak i klienta)
- prywatności (poprzez wykorzystanie szyfrowanego połączenia)
- integralności przesyłanych danych (poprzez wykorzystanie sum kontrolnych)

SSL jest protokołem scentralizowanym, wykorzystuje instytucje certyfikujące (Certyfing Authorities) Wykorzystuje trzy rodzaje certyfikatów:

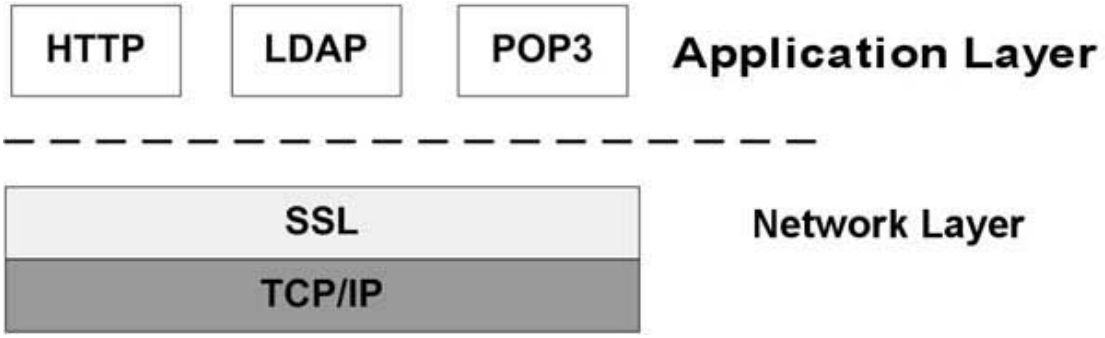
- certyfikat instytucji certyfikujacej - zestaw informacji określających tożsamość danej instytucji (CA) , obecność certyfikatu na certyfikacie serwera świadczy, że CA poświadcza tożsamość serwera
- certyfikat serwera - zestaw informacji określających tożsamość serwera
- certyfikat użytkownika - wraz z certyfikatem CA poświadcza tożsamość klienta

Obecnie istnieją dwie specyfikacje:

* SSL 2.0
* SSL 3.0

3.2 Budowa SSL

SSL jest warstwą ulokowaną pomiędzy protokołem TCP/IP a takimi protokołami jak HTTP.



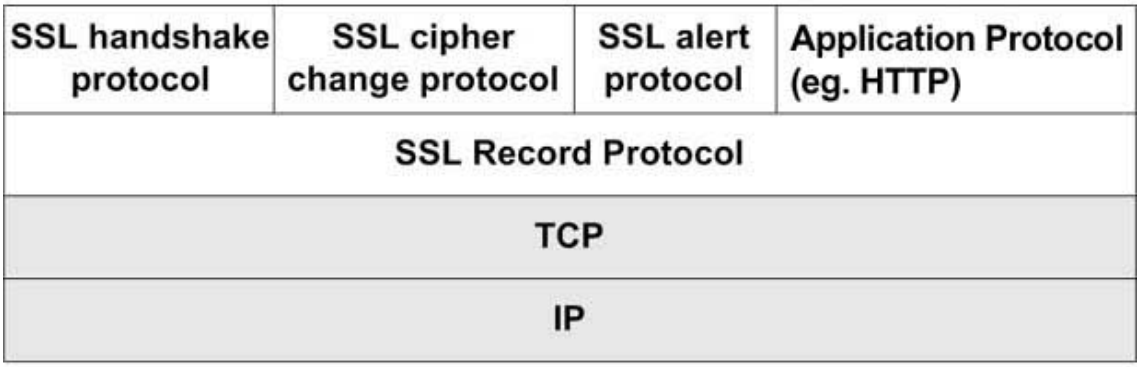
Rys 6. SSL Umiejscowienie

W gruncie rzeczy na SSL składa się kilka protokołów, które można pogrupować na dwie warstwy:

* warstwa odpowiadająca za realizację zadań związanych z bezpieczeństwem oraz integralnością danych - SSL Record Protocol

* warstwa odpowiadająca za nawiązanie bezpiecznego połączenia SSL; składają się na nią następujace protokoły

    - SSL Handshake Protocol
    - SSL ChangeCipher SpecPprotocol
    - SSL Alert Protocol



Rys 7. SSL Składowe

* SSL Record Protocol - jest wykorzystywany do przesyłania danych czy komunikatów w ramach sesji.

* SSL Handhake Protokol - najbardziej złożona część protokołu SSL. Tu realizowane jest nawiązanie połączenia, uzgodnienie szczegółów dotyczących sesji takich jak klucze czy algorytmy szyfrujace używane w trakcie sesji. SSL Handhake Protokol odpowiada także za wzajemną autoryzację komunikujących sie stron i ustalenie pozostałych parametrów.

* SSL ChangeCipher SpecProtocol - najprostszy z wszystkich protokołów; wykorzystywany do przesłania w obie strony komunikatu o treści 1. Przesłanie takich komunikatów kończy etap uzgadniania szczegółów sesji SSL.

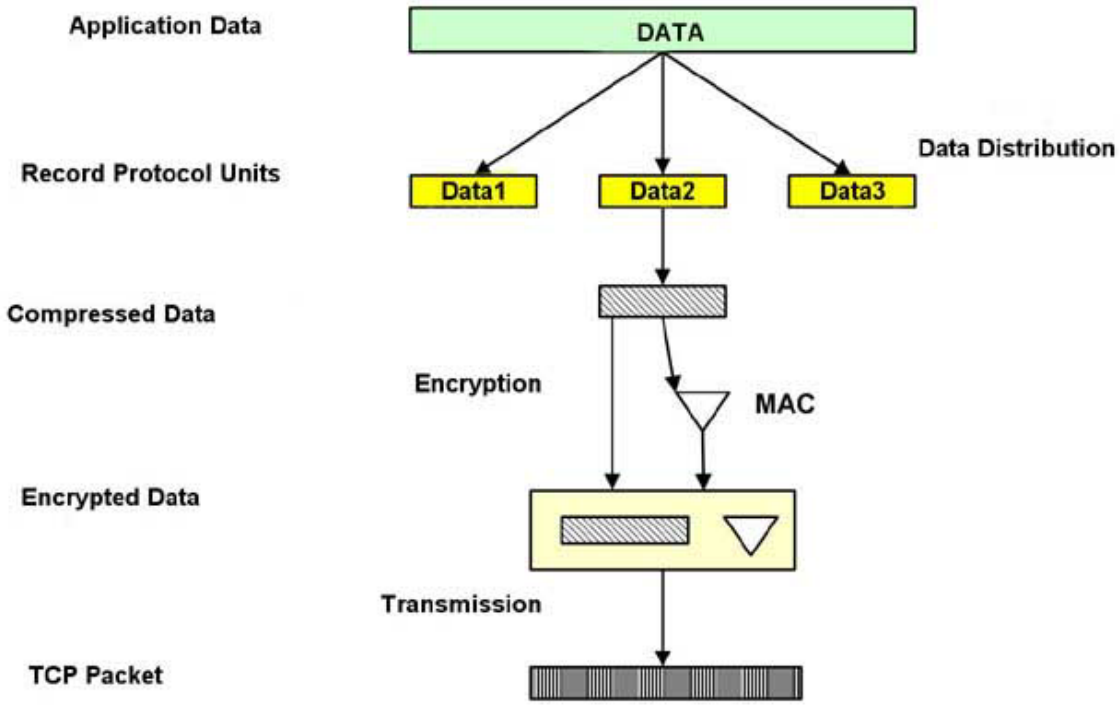
* SSL Alert Protocol - wykorzysywany do przesyłania danych i komunikatów związanych z przesyłaniem danych i funkcjonowaniem protokołu. Taki komunikat składa sie z dwóch bajtów. Wartość pierwszego może oznaczać status "warning"(1) lub "fatal"(2). Drugi bajt to kod błędu mogącego wystąpić poczas sesji SSL. (Przesłanie komunikatu o statusie "fatal"prowadzi do natychmiastowego zakończenia sesji.)

W dalszej części omówione są dokładniej dwie najważniejsze składowe SSL - SSL Record Protocol oraz SSL Handhake Protokol

3.3 SSL Record Protocol

Działalnie tego protokołu składa sie z następujących kroków.

1. Pobranie danych (np. dane aplikacji) które mają zostać przesłane.
2. Podział danych.
3. Kompresja danych przy użyciu uzgodnionego wcześniej algorytmu.
4. Szyfrowanie dokumentu oraz wyliczenie wartości MAC (wartość funkcji haszującej na danych).
5. Przesłanie tak zbudowanego pakietu.



Rys. 8 Schemat działania SSL Record Protocol.

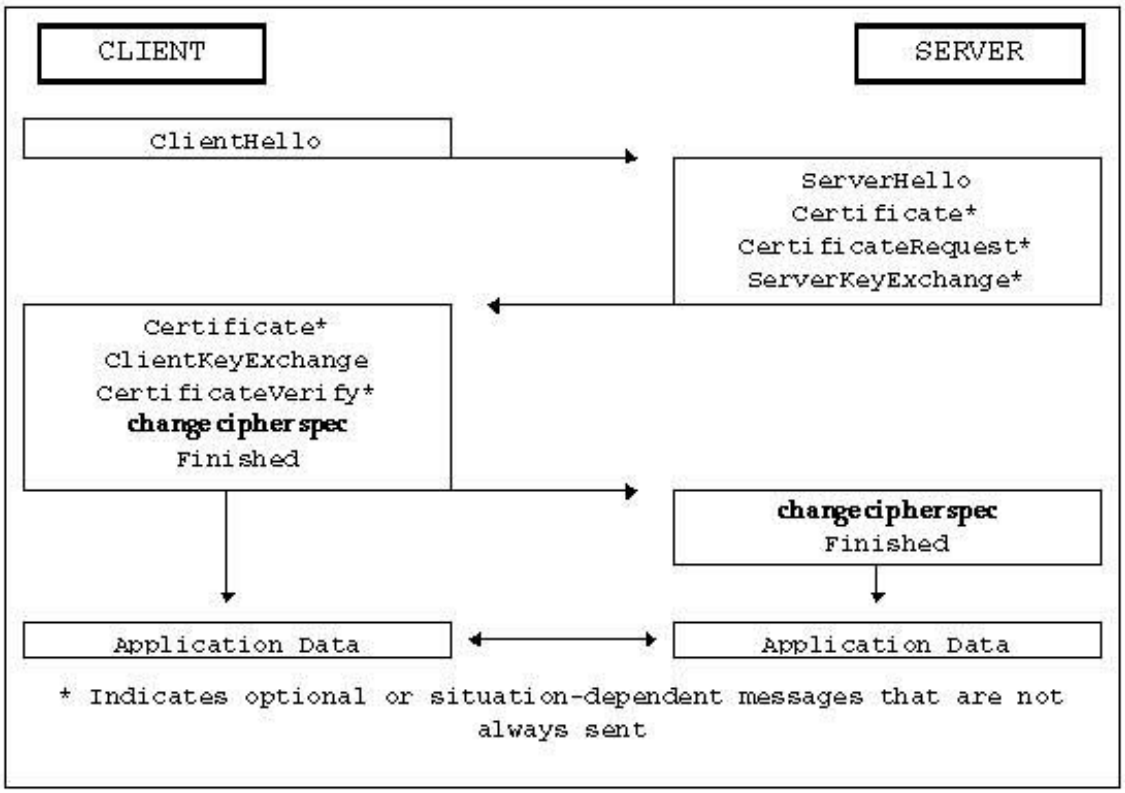
3.4 SSL Handshake Protocol

SSL Handshake Protocol odpowiedzialny jest za nawiązywanie połączenia pomiędzy klientem a serwerem.

1. Klient wysyła do serwera komunikat client hello; komunikat ten zawiera takie informacje jak:
    * ID sesji (0 w przypadku nawiązywaniu nowego połączenia lub ID tej sesji, której ustawienia mają być wykorzystane przy tym połączeniu)

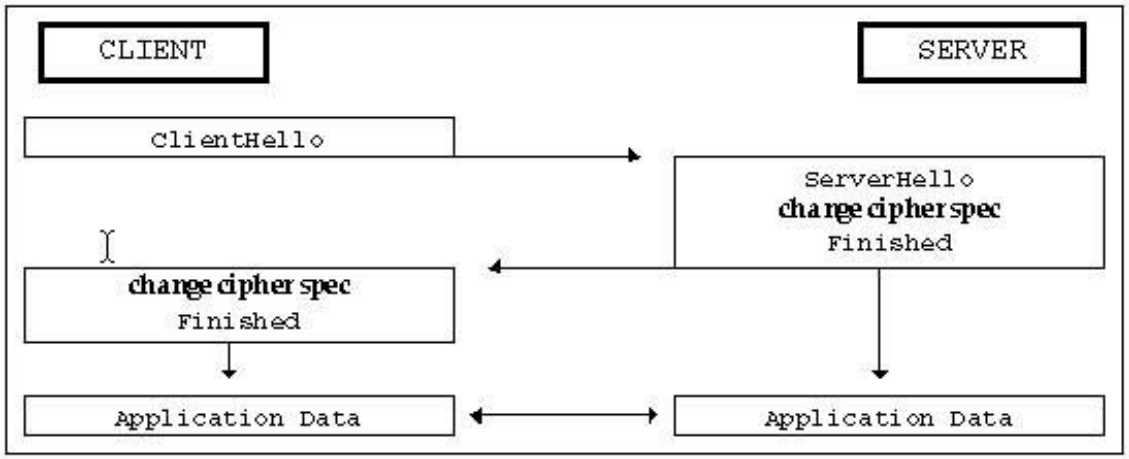
    * lista algorytmów szyfrujących znana klientowi (uporządkowana według jego preferencji)
    * lista algorytmów kompresujących znanych klientowi (uporządkowana według jego preferencji); jeśli serwer nie zna żadnego z tych algorytmów połączenie sie nie powiedzie
    * najwyższa wersja SSL wspierana przez klienta

2. Serwer odpowiada komunikatem server hello; komunikat ten zawiera takie informacje jak:
    * ID sesji
    * wybrany spośród listy zaproponowanej przez klienta algorytm szyfrujący
    * wybrany spośród listy zaproponowanej przez klienta algorytm kompresji
    * najniższa wspierana przez serwer wersja SSL
3. Serwer wysyła swój certyfikat
4. Serwer wysyła server key exchange komunikat
5. Serwer może wysłać prośbe o certyfikat klienta
6. Serwer wysyła wiadomość server hello done kończąc pierwszą fazę komunikacji.
7. Jeśli serwer poprosił o certyfikat klienta ten odpowiada komunikatem certificate message lub (jeśli nie ma certyfikatu) komunikatem no certificate.
8. Klient wysyła komunikat client key exchange.
9. Jeśli jest to konieczne wysyłana jest także wiadomość certificate verify w celu zweryfikowania ceryfikatu.
10. W następnym kroku klient oraz serwer wymieniają sie komunikatami change cipher spec (SSL ChangeCipher SpecPprotocol) oraz kończą etap uzgadniania szczegółów komunikacji komunikatem finish.
11. Dalej ma miejsce komunikacja, w której przesyłane są właściwe dane przy wykorzystaniu uzgodnionych kluczy oraz algorytmów szyfrujących oraz kompresujących.



Rys. 9 Schemat działania SSL Handshake Protocol.

Ten schemat odzwierciedla komunikację przy nawiązywaniu nowego połączenia. W przypadku wznowienia sesji ogranicza się ona tylko do wymiany komunikatów client hello, server hello oraz change cipher spec.



Rys. 10 Wznawianie sesji (SSL Handshake Protocol).

4 SSH

4.1 Wprowadzenie

SSH (Secure Shell) to narzędzie umożliwiające bezpieczną pracę na zdalnym komputerze. SSH został wprowadzony w miejsce takich narzędzi jak telnet czy rlogin, które zaprojektowne w innej rzeczywistości przestały wraz z ogromnym rozwojem Internetu spełniać bezpiecznie swoje zadania. Obecnie dostępne są zarówno darmowe jak i komercyjne wersje SSH na większość dostępnych platform. Dla systemów Unix/Linux najbardziej popularnym oprogramowaniem jest pakiet OpenSSH. Pierwsze prace nad SSH skończyły sie w 1995 roku kiedy to powstała pierwsza wersja (SSH1). Obecnie wykorzystywana jest bezpieczniejsza i pozbawiona wielu niedociągnięć swojej poprzedniczki wersja SSH2. SSH podobnie jak SSL także wykorzystuje mechanizmy kryptograficzne związane z kluczem publicznym.

SSH oferuje zabezpieczenie w sytuacji:

* IP spoofing

* IP source routing

* DNS spoofing

* gdy przesyłane dane są zatrzymywane przez pośredniczącego hosta

Oczywiście SSH zawiedzie w przypadku gdy atakującemu uda się zdobyć kontrolę nad komputerem oferującym usługę SSH

4.2 Schemat działania wersji SSH1

Działanie SSH1 opiera sie na założeniu że host akceptujacy połączenia ssh ma swój unikalny klucz RSA dla celów identyfikacji. Jest to tzw. host key. Długość tego klucza wynosi 1024 bity. Dodatkowo przy uruchomienia serwera SSH generowany jest kolejny klucz (server key) o długości 768 bitów . Ten klucz jest odtwarzany po stałych okresach czasu (zazwyczaj jest to godzina) o ile klucz był używany. Każdorazowo gdy klient łączy sie z serwerem otrzymuje parę kluczy. Na podstawie host key klient sprawdza tożsamość serwera (sprawdza czy klucz, który otrzymał znajduje się w jego bazie - jeśli nie to go tam zapisuje). W następnym kroku generowany jest w sposób losowy klucz 256 bitowy (session key). Jest on następnie kodowany przy wykorzystaniu host key oraz server key oraz wysyłany do serwera. Serwer rozkodowuje przy pomocy swoich prywatnych kluczy taki komunikat i od tej pory session key będzie wykorzystywany do symetrycznego kodowania dalszej części sesji. Do kodowania transmisji przy użyciu klucza symetrycznego wykorzystywane są algorytmy Blowfish oraz 3DES . Wyboru algorytmu dokonuje klient wybierając algorytm z tych proponowanych przez serwer. Kolejnym etapem jest uwierzytelnienie użytkowników. Wykorzystywane tu mechanizmy nie ograniczają sie tylko do wykorzystania hasła. Dostępne są następujące sposoby:

* jeśli nazwa maszyny, z której klient się loguje znajduje się w katalogu /etc/hosts.equiv lub /etc/shosts.equiv na zdalnej maszynie i nazwa użytkownika jest taka sama na obu komputerach to użytkownik może sie zalogować

* jeśli istnieją pliki .rhosts lub .shosts w katalogu domowym użytkownika na zdalnej maszynie i zawierają nazwe maszyny z której klient się łączy i nazwę użytkownika na tej maszynie to użytkownik również może sie zalogować

Jako że te dwie metody nie są bezpiecznie zwykle nie są wykorzystywane. Kolejną metodą uwierzytelniania jest mechanizm wykorzystujący klucz publiczny. W pliku znajduje się lista kluczy publicznych które mają zgodę na połączenie. Klient łącząc sie z serwerem podaje którego klucza chciałby użyć do uwierzytelnienia jego wiarygodności. Jeśli taki klucz jest w wspomnianym pliku serwer generuje losową wartość i szyfruje ją tym kluczem. Jeśli po odkodowaniu przez klienta otrzyma on tą samą wartość to jego wiarygodność zostaje potwierdzona. Użytkownicy mogą generować własne klucze używając programu ssh-keygen. Ich klucze prywatne przechowywane są w odpowiednich plikach w ich katalogu domowym.
Aby opisany sposób uwierzytelniania mógł być wykorzystany konieczne jest skopiowanie klucza publicznego do właściwego pliku w katalogu domowym na zdalnej maszynie.
Kolejną metodą uwierzytelniania jest wykorzystanie programu agenta. Mechanizm ten jest także wspierany w wersji SSH2. Jeśli użytkownik nie został uwierzytelniony żadną z wymienionych metod konieczne jest podanie hasła (oczywiście przesyłanego w postaci zakodowanej)

4.3 Schemat działania wersji SSH2

SSH2 w odróżnieniu od SSH1 nie wykorzystuje server key. Wykorzystywane do kodowania transmisji algorytmy wykorzystujące klucz symetryczny to :

* 128 bit AES

* Blowfish

* 3DES

* CAST128

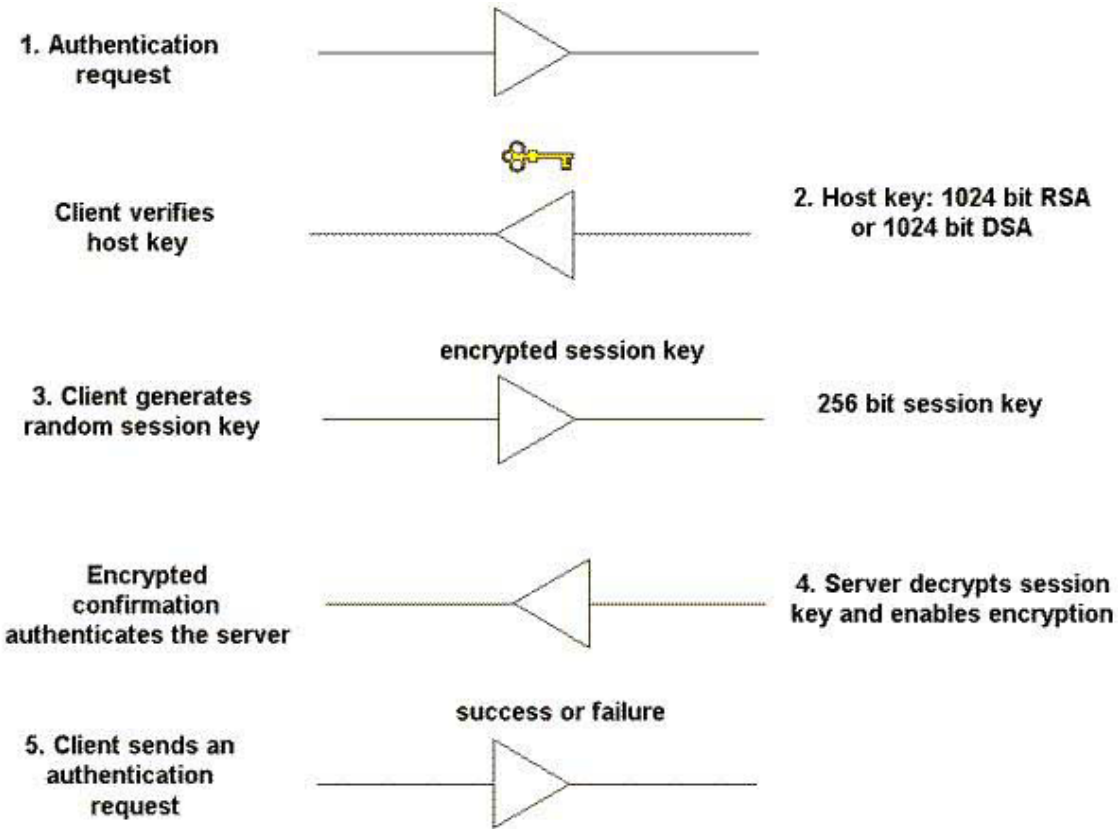
* Arcfour

* 192 bit AES

* 256 bit AES

Dodatkowo do kontrolowania integralności przesyłanych danych SSH2 wykorzystuje takie algorytmy haszujace jak: SHA1 czy MD5.

Autoryzacja maszyn przedstawiona jest na poniższym rysunku.



Rys. 1 1 SSH2 Autoryzacja

1. Klient nawiązuje połączenie z serwerem.
2. Serwer przedstawia klientowi swój publiczny host key. Klient sprawdza swoją bazę danych w poszukiwaniu tego klucza. (Jeśli jest to pierwsze połączenie z danym serwerem to nowy klucz jest dodawany). Klient jest informowany jeśli w bazie danych danemu serwerowi odpowiada inny klucz.
3. Klient generujue losową 256 bitową liczbe, która zostanie wykorzystana do zakodowania transmisji oraz dokonuje wyboru algorytmu kodowania. Zakodowany klucz jest przesyłany do serwera.
4. Serwer dokonuje rozkodowania klucza i wysyła klientowi potwierdzenie.
5. Klient wysyła prośbę o uwierzytelnienie.

Sposobu uwierzytelniania w SSH2 są podobne do tych wykorzystanych w SSH1 (większość różnic sprowadza sie do różnych nazw plików czy wykorzystania innych algorytmów). W przeciwieństwie do SSH1 tutaj klient wysyła do serwera wiadomość zaszyfrowaną swoim kluczem prywatnym a serwer sprawdza czy odpowiedni klucz publiczny znajduje sie wśród kluczy z prawem dostępu.

SSH2 dopuszcza możliwość wykorzystania do uwierzytelniania Kerberosa, cartyfikatów X.509 czy Smart Cards.

5 Bibliografia

1. http://www.signet.pl/pomoc/pki.html
2. http://integrator.solidex.pl/?php-wid=58-php-aid=9
3. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/default.asp
4. http://www.tldp.org/HOWTO/Smart-Card-HOWTO/index.html
5. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/default.asp
6. http://wp.netscape.com/eng/ssl3/3-SPEC.HTM
7. http://www.windowsecurity.com/pages/article.asp?id=440
8. http://www.windowsecurity.com/pages/article.asp?id=439
9. http://www.sans.org/rr/encryption/intro-SSH.php