Protokół SSL/TLS

Patryk Czarnik

Bezpieczeństwo sieci komputerowych – MSUI 2010/11

Algorytmy wymiany klucza – motywacja

- Kryptografia symetryczna efektywna
- Ale wymagana znajomość tajnego klucza przez obie strony
- W internecie trudno zapewnić bezpieczny kanał dystrybucji klucza
- Zamiast tego kryptograficzny algorytm wymiany klucza

Ustalona może zostać tylko mała porcja tajnych danych, która dopiero potem posłuży obu stronom do wygenerowania dłuższego klucza (w oparciu o losowe, ale nie tajne dane oraz bezpieczne funkcje haszujące).

Algorytm Diffiego-Hellmana – wstęp

- Pozwala dwóm stronom na ustalenie tajnego klucza w niezabezpieczonej sieci.
- Nie wymaga znajomości żadnych tajnych informacji ani obecności zaufanej „trzeciej strony”.
- Autorzy: W. Diffie i M. Hellman, w oparciu o wcześniejsze badania R. Merkle
  - M. J. Williamson niezależnie (wcześniej) odkrył tę samą metodę, ale jego odkrycie pozostało tajne.
- Wersja podstawowa oraz rozszerzenia umożliwiające m.in. uwierzytelnienie stron.

Algorytm Diffiego-Hellmana – opis

1. Alicja i Bartek ustalają (dużą) liczbę pierwszą p oraz generator g. Te liczby mogą być znane atakującemu.
2. Dalej wszystkie operacje odbywają się w grupie „modulo p”.
3. Alicja losuje liczbę a, a Bartek liczbę b.
4. Alicja wysyła do Bartka g^a, a Bartek do Alicji g^b.
5. Alicja oblicza $(g^b)^a$, a Bartek $(g^a)^b$.
6. Oboje obliczyli tę samą liczbę $g^{a \times b}$.
7. Atakujący, aby poznać tę liczbę, musiałby rozwiązać tzw. **problem Diffiego-Hellmana**, uznawany za obliczeniowo trudny, równie trudny jak problem logarytmu dyskretnego (ale bez dowodu :).

Wersje algorytmu Diffiego-Hellmana (1)

Wersja **anonimowa**

- Zgodna z wcześniej opisanym algorytmem
- Brak uwierzytelnienia
- Nie odporna na atak typu „człowiek w środku”

Wersja **z hasłem** (*password-authenticated*)

- Wymagana wcześniejsza znajomość tajnego hasła przez obie strony
- Pewien parametr publiczny (zwykle g) tworzony na podstawie hasła i nie przesyłany
- Mało popularny, w szczególności nie używany w TLS

Wersje algorytmu Diffiego-Hellmana (2)

Wkorzystanie kryptografii klucza publicznego

Wersja **ustalona** (*fixed*)

- Certyfikat serwera, a opcjonalnie także klienta, zawiera publiczne parametry D-H, podpisane przez CA
- Może to skutkować tym samym kluczem w kolejnych sesjach

Wersja **efemeryczna** (*ephemeral*)

- Klucz sesji zmienny
- Motywacja – aby przechwycenie klucza (innymi środkami) nie dało możliwości odszyfrowania przeszłej ani przyszłej komunikacji
- Uwierzytelnienie zapewnione poprzez podpisywanie kluczami prywatnymi i weryfikację kluczami publicznymi z certyfikatów

Wymiana klucza oparta o klucz publiczny (np. RSA)

1. Bartek zna klucz publiczny Alicji.
2. Bartek generuje losowy klucz sesji, szyfruje go kluczem publicznym Alicji i wysyła.
3. Tylko Alicja może odszyfrować klucz sesji.
4. Ten scenariusz może być wzbogacony o uwierzytelnienie Bartka (np. podpisanie komunikatu kluczem prywatnym Bartka, co Alicja zweryfikuje jego kluczem publicznym).

Historia

SSL (*Secure Socket Layer*) i **TLS** (*Transport Layer Security*) to protokoły bezpiecznej komunikacji w sieciach komputerowych. Historia:

- SSL został opracowany przez Netscape. Głównym zastosowaniem miała być bezpieczna komunikacja przeglądarki internetowej z serwerem (HTTPS).
- 1996 – SSL 3.0, publiczny standard Netscape, stosowany do dzisiaj.
- 1999 – TLS 1.0, oparty o SSL 3.0 standard IETF (RFC 2246).
- 2006 – TLS 1.1 (RFC 4346), obecnie obowiązująca wersja.
- Trwa opracowywanie TLS 1.2.

Architektura i zastosowania

- Formalnie protokół znajduje się w warstwie aplikacji (najwyższej) modelu TCP/IP oraz w warstwie prezentacji (6/7) modelu OSI.
- Protokół działa nad TCP (lub potencjalnie innym „pewnym” protokołem warstwy transportowej) a pod protokołem aplikacji (np. HTTP).
- Implementacja SSL/TLS może być częścią aplikacji użytkownika (inaczej niż IPsec).
- Możliwe jest wykorzystanie SSL/TLS w protokołach innych niż HTTP (np. Telnet, SMTP, FTP).
- Do zastosowań należą także:
  - tunelowanie dowolnej komunikacji opartej o TCP,
  - budowa VPN (nie ma takich problemów z firewallami i NAT jak przy IPsec); przykładowa implementacja: OpenVPN.

Z punktu widzenia programisty

- Biblioteki do obsługi SSL i TLS ogólnie dostępne (np. OpenSSL)
- Abstrakcja „bezpiecznych gniazd”
  - wewnętrznie gniazda otoczone warstwą bezpieczeństwa
  - z zewnątrz dostępne poprzez „zwykły” interfejs gniazd
- Możliwość zastosowania istniejącego kodu operującego na „zwykłych” gniazdach do gniazd bezpiecznych
- Plus oczywiście konfiguracja parametrów bezpieczeństwa, ale zwykle wystarczy podczas tworzenia

Dostępne metody wymiany klucza

- Wykorzystanie kryptografii klucza publicznego (RSA lub DSA). Wymaga podania certyfikatu.
- Kilka postaci algorytmu Diffiego-Hellmana:
  - ustalony,
  - efemeryczny,
  - anonimowy (bez uwierzytelniania, tylko szyfrowanie).
- Fortezza:
  - system oparty o karty chipowe,
  - stosowany m.in. w instytucjach rządowych USA,
  - wycofany w wersji TLS 1.0.

Dostępne algorytmy kryptograficzne

- Szyfrowanie: RC2, RC4, DES, 3DES, DES40, IDEA, Camellia.
- Hasz: MD2, MD4, MD5, SHA-1.

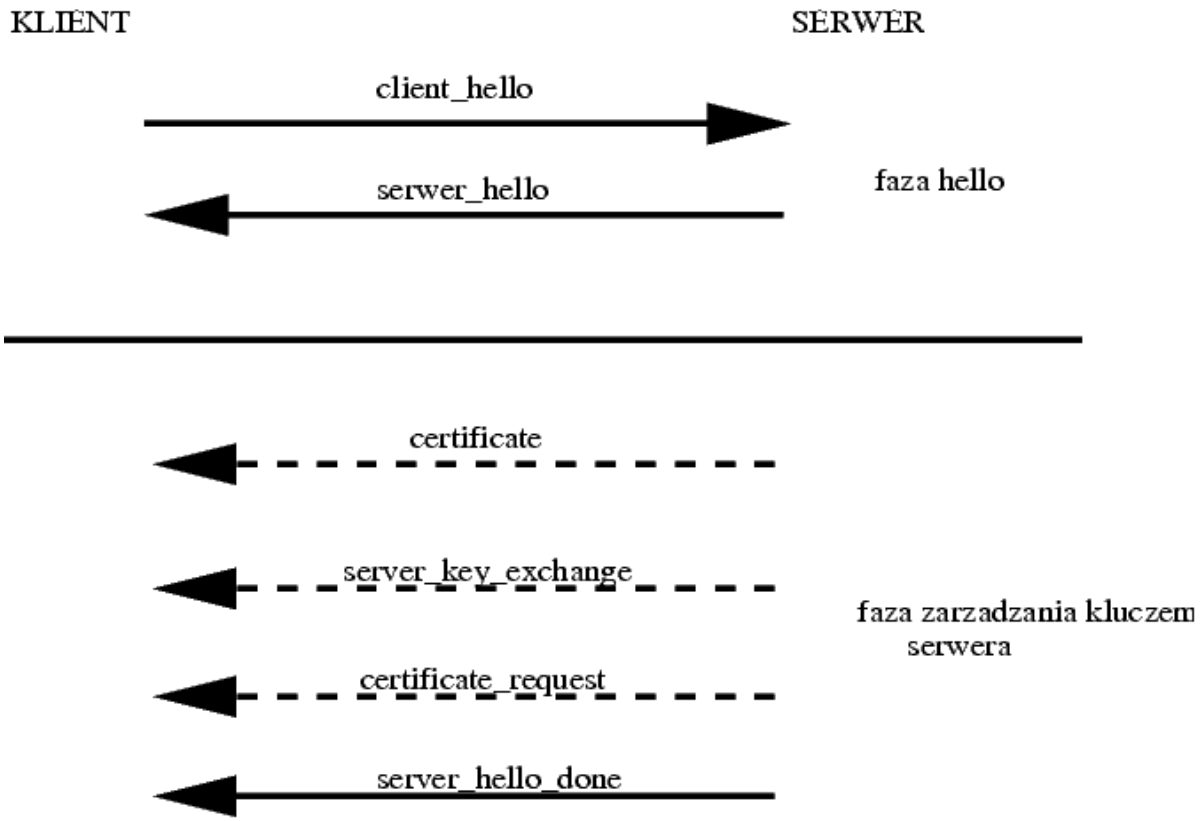
Podprotokoły SSL

- **Record Protocol** – przesyłanie danych w rekordach. Rekord posiada identyfikator typu zawartości. Rekord może być skompresowany, zaszyfrowany, opatrzony kodem MAC.
- **Handshake Protocol** – inicjalizajca, negocjacja kluczy i transformacji kryptograficznych.
- **Change Cipher Spec** – zmiana ustalonych parametrów kryptograficznych.
- **Alert Protocol** – protokół powiadamiania o błędach.

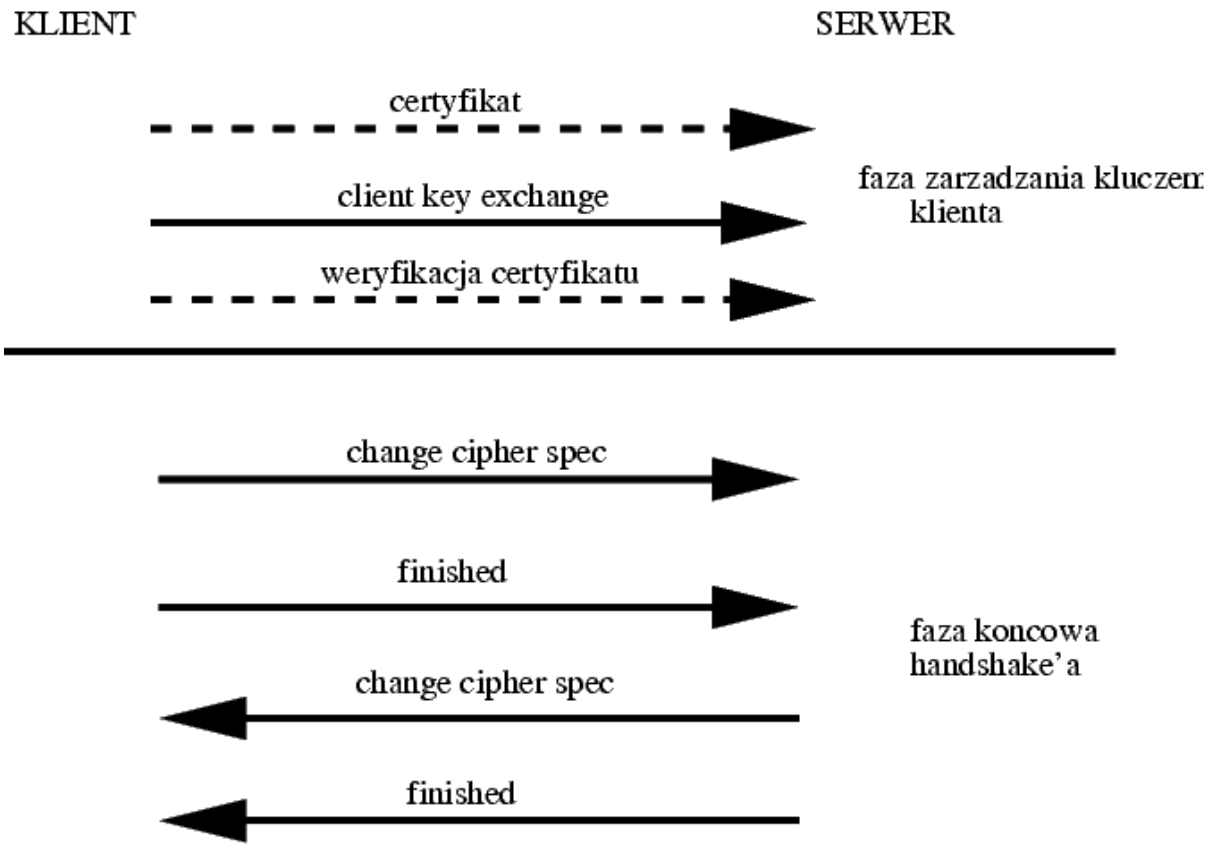
Rzut oka na działanie SSL

- Serwer czeka na porcie TCP 443.
- Klient łączy się z gniazdem TCP i wiadomością `ClientHello` rozpoczyna fazę Handshake (szczegóły dalej).
- Także wiadomości protokołu Handshake wymieniane są zgodnie z Record Protocol (w rekordach z typem zawartości 22).
- Faza Handshake kończy się ustaleniem parametrów protokołem Change Cipher Spec.
- Dalej protokołem Record wymieniane są zaszyfrowane komunikaty wyższej warstwy.
- W razie potrzeby może nastąpić wymiana wiadomości Change Cipher Spec lub Alert.

Handshake protocol – schemat



Handshake protocol – schemat (2)



Zawartość `ClientHello`

- Wersja SSL (najwyższa jaką obsługuje).
- Wartość losowa (pieczątka czasowa + 28 losowych bitów).
- Identyfikator sesji (0 – rozpoczęcie nowej sesji).
- Zestaw szyfrów (rodzaj i parametry algorytmu, klucz sesji) i metod wymiany klucza.
- Lista dostępnych metod kompresji.

Zawartość `ServerHello`

Ten sam zestaw danych co w `ClientHello`, z tym że:

- Wartości losowe są niezależnie generowane przez klienta i serwer
- Jeśli klient zażądał nowej sesji (0), serwer generuje i podaje nowy identyfikator sesji
- Zestaw szyfrów i metod kompresji: klient proponuje, serwer wybiera
- Zazwyczaj przesyłany certyfikat serwera

Komunikat `server_key_exchange`

- Komunikat serwera w obranym algorytmie wymiany klucza. Może nie być potrzebny, gdy wystarcza certyfikat.

Komunikat `certificate_request`

- Serwer może zażądać certyfikatu od klienta.
- Żądanie może zawierać:
  - typ certyfikatu (RSA, DSS, rozmaite wersje DH),
  - listę CA akceptowanych przez serwer.

Komunikat `server_done`

- Oznacza, że serwer wysłał wszystkie informacje i żądania.
- Klient może teraz:
  - zweryfikować certyfikat serwera,
  - jeśli serwer tego zażądał – wysłać własny certyfikat lub wiadomość `no_certificate alert`.
  - wysłać swoją wiadomość `client_key_exchange`.

Komunikat `client_key_exchange`

- W zależności od wybranego sposobu wymiany klucza sekret zaszyfrowany kluczem publicznym serwera lub druga wiadomość w algorytmie DH.

Faza końcowa

- Klient wysyła `change_cipher_spec`.
- „Od tej pory będę korzystał z wynegocjowanych parametrów i klucza”.
- Klient wysyła `finished` podpisany przy pomocy wynegocjowanych kluczy.
- Serwer odpowiada w taki sam sposób.

Zmiany w TLS w stosunku do SSL 3.0

- Zrezygnowano z wymiany klucza Fortezza.
- Wykorzystuje się HMAC (konstrukcja tworząca bezpiecznego MAC-a z bezpiecznej funkcji haszującej) opartego o MD5/SHA-1.
- TLS umożliwia stosowanie trybu wstecznej kompatybilności z SSL 3.0, poza szczególnymi przypadkami, gdy druga strona jest bardzo ograniczona (np. umie używać tylko Fortezzy).

Idea instytucji uwierzytelniających (CA)

- Kryptografia klucza publicznego oparta o zaufanie do klucza publicznego
- PKI (*Public Key Infrastructure*) to zestandaryzowana idea zapewniania zaufania do certyfikatów w oparciu o dziedziczenie zaufania w strukturze drzewiatej
- CA (*certificate authority*) to instytucja obdarzona „autorytetem”, posiadająca własny certyfikat
- CA podpisuje certyfikaty CA niższego rzędu
- CA podpisuje certyfikaty stron
- Strona A może mieć zaufanie do strony B, jeśli tylko ma zaufanie do CA znajdującego się wśród przodków B w drzewie certyfikatów

Podstawowe operacje w PKI

- Rejestracja – użytkownik wnioskuje o certyfikat
- Certyfikacja – wydanie certyfikatu
  - Generowanie pary kluczy zwykle po stronie użytkownika, CA umieszcza klucz publiczny w certyfikacie i podpisuje
- Odnawianie certyfikatów – ponad pierwotnie ustalony termin ważności
- Unieważnianie certyfikatów – listy CRL (*Certificate Revocation List*)
- Wzajemna certyfikacja – różne CA uwierzytleniają się nawzajem aby zwiększyć poziom zaufania do siebie

Sprawdzanie wiarygodności serwerów

- Certyfikaty serwera podpisane przez CA znane przeglądarce:
  - przeglądarka weryfikuje certyfikat korzystając z klucza publicznego CA
  - przeglądarka sprawdza czy certyfikat nie występuje na liście CRL
- Certyfikaty podpisane przez serwer:
  - przeglądarka pyta użytkownika czy wierzyć certyfikatowi

Zaufanie do certyfikatów

- Serwera:
  - publicznie dostępne serwery powinny posiadać certyfikaty podpisane przez niezależne instytucje, aby zapewnić że są tymi, za kogo się podają,
  - samocertyfikacja (firma podpisuje certyfikaty swoim serwerom) powinna wystarczyć pracownikom firmy.
- Klienta:
  - szyfrowanie i uwierzytelnianie serwera nie ogranicza użytkownikom dostępu do serwera; czasem może więc być potrzebne uwierzytelnianie klientów,
  - dostępne w SSL/TLS, ale rzadko wykorzystywane.

Zawartość certyfikatu

- SSL/TLS używa certyfikatów X.509 (w wersji TLS 1.2 planowana także obsługa certyfikatów PGP)
- Certyfikat może zawierać m.in.:
  - nazwę (domenową) strony
  - nazwa Instytucji Autoryzującej (CA)
  - klucz publiczny strony
  - rodzaj (algorytm i parametry) klucza publicznego
  - datę ważności
  - parametry algorytmów kryptograficznych użytych do podpisu
  - podpis CA całego certyfikatu

Certyfikaty użytkowników czy hasła?

- Zalety certyfikatów:
  - certyfikaty można bezpiecznie wysyłać przez sieć (haseł nie, przynajmniej niezaszyfrowanych),
  - certyfikatów nie trzeba pamiętać w głowie :),
  - są silniejszą (kryptograficznie) metodą uwierzytelnienia.
- Wady certyfikatów:
  - plik z certyfikatem można zgubić, uszkodzić,
  - pliki z certyfikatem są przypisane do komputera (nazwa hosta),
  - przydatne są hasła do certyfikatów – ale wtedy trzeba je pamiętać jak zwykłe hasła.