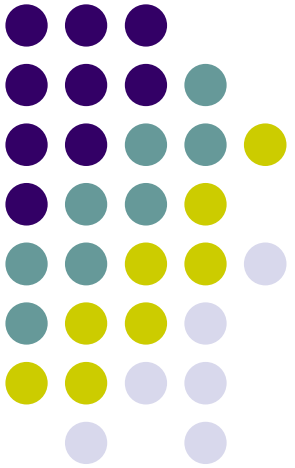
Bezpieczeństwo systemów informatycznych

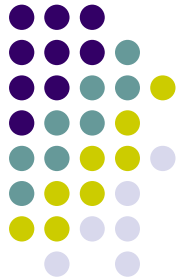
Wykład 4

Protokół SSL

Tomasz Tyksiński, WSNHiD

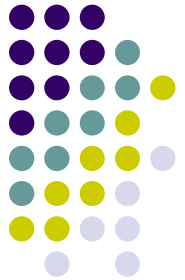


Rozkład materiału



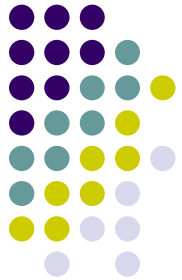
1. Podstawy kryptografii
2. Kryptografia symetryczna i asymetryczna
3. Podpis elektroniczny i certyfikacja
4. Bezpieczeństwo warstwy transportowej
5. Bezpieczeństwo warstwy sieciowej

Protokoły SSL i TLS



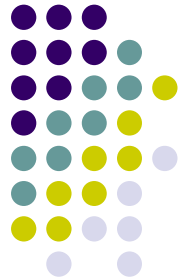
- Wprowadzenie
- Wersje protokołów SSL/TLS
- Opis protokołów składowych
- Przesyłane wiadomości
- Etapy nawiązywania połączenia
- Bezpieczeństwo

Wprowadzenie



- *Secure Sockets Layer* (SSL) oraz *Transport Layer Security* (TLS) to protokoły umożliwiające ustanowienie bezpiecznego połączenia między dwiema maszynami.
- Musimy zatem zagwarantować poufność, integralność przesyłanych danych oraz uwierzytelnienie punktów końcowych.

Protokół SSL

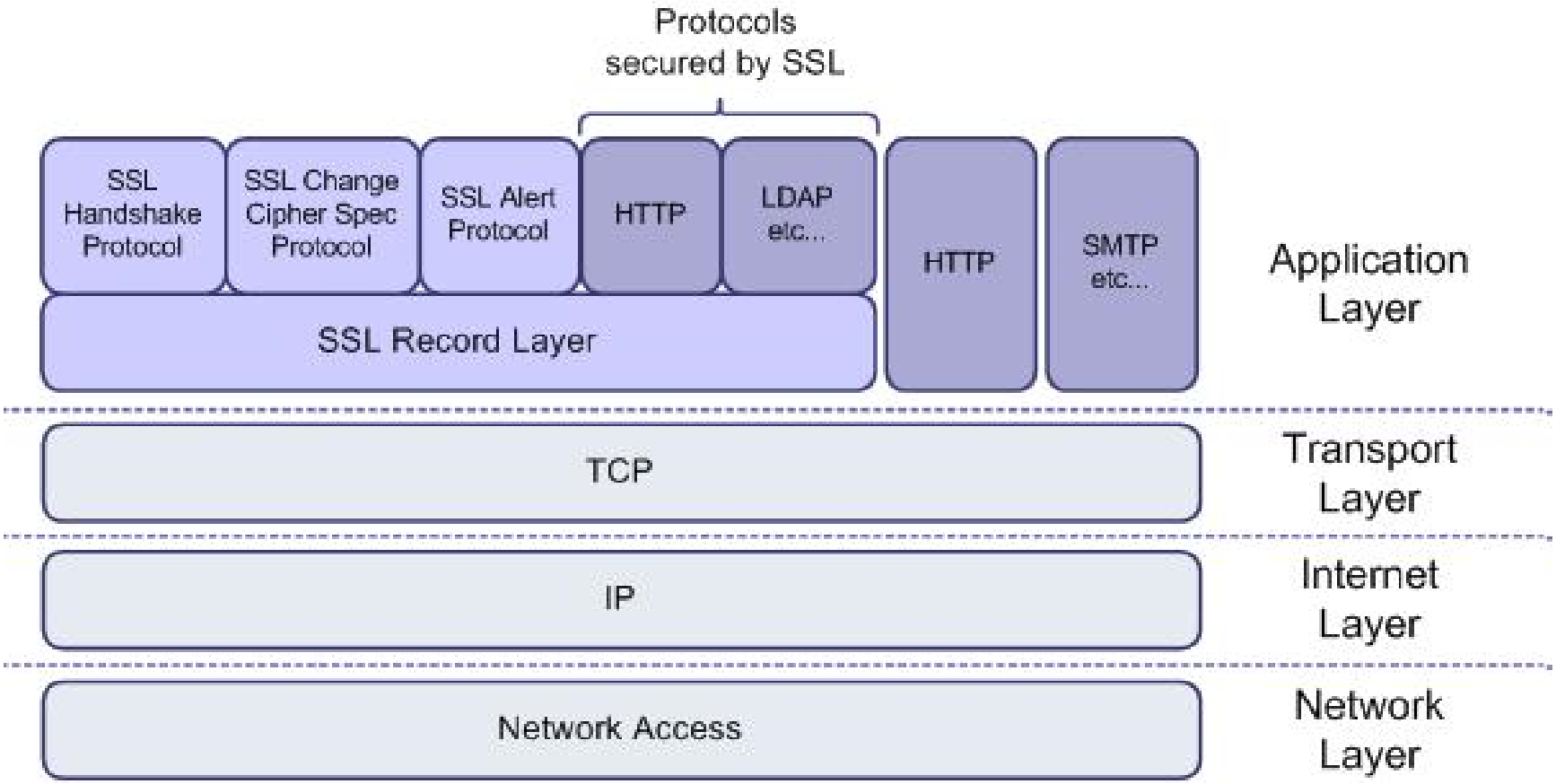


- Poufność uzyskiwana jest za pomocą kryptografii symetrycznej i asymetrycznej.
- Integralność danych jest sprawdzana za pomocą funkcji MAC.
- Uwierzytelnianie wykorzystuje kryptografię asymetryczną, funkcje haszujące oraz certyfikaty.
- Protokół ogólnie zabezpiecza warstwy sieciowe ponad warstwą transportową.

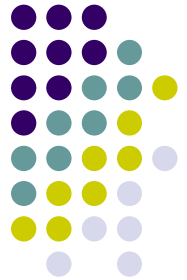
Model warstwowy

	OSI	TCP/IP
7	Aplikacji	Aplikacji (Telnet, FTP)
6	Prezentacji	Sesji (RPC)
5	Sesji	Transportowa (gniazda)
4	Transportowa	TCP UDP
3	Sieci	Sieciowa (IP)
2	Łącza danych	Protokołów fizycznych (Ethernet)
1	Fizyczna	Medium transmisji (kabel)

Model warstwowy

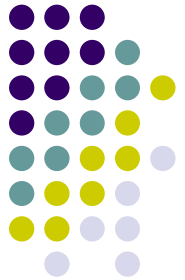


Protokół SSL



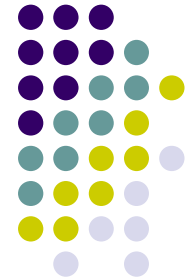
- SSL/TLS jest protokołem typu klient/serwer.
- Stosowany jest do zabezpieczenia innych protokołów klient/serwer, przykładowo HTTPS, POP3S czy IMAP4S. Wówczas przy uwierzytelnianiu sprawdzana jest tylko strona serwera przez klienta.
- Oznacza to, że web serwer lub serwer pocztowy musi mieć certyfikat, natomiast klient nie musi go posiadać.
- W protokołach SSL i TLS można również wykorzystać również certyfikat klienta, na żądanie serwera. Jest to wygodne przy autoryzacji dostępu do usług.
- Jest także możliwość anonimowej wymiany klucza, wówczas serwer nie wysyła certyfikatu uwierzytelniającego. Oczywiście nie daje to żadnej gwarancji bezpieczeństwa.

Protokół SSL



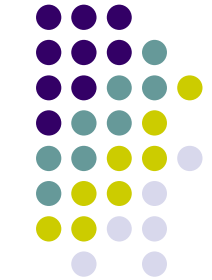
- Przy sprawdzaniu certyfikatów należy oczywiście sprawdzić: datę wygaśnięcia, nazwę podmiotu zapisaną w certyfikacie, podpis elektroniczny urzędu poświadczającego oraz czy certyfikat nie został zawieszony (listy CRL).

Wersje protokołów SSL



- SSL 1.0 – opracowana przez Netscape w 1994, nigdy nie opublikowana ze względu na problemy z bezpieczeństwem.
- SSL 2.0 – wydana w lutym 1995, ale również posiadała błędy i szybko została uaktualniona.
- SSL 3.0 – wydana w 1996, opracowana przez Paula Kochera wraz z pracownikami Netscape - Philem Karltonem, Alanem Freierem, bardzo popularna, choć dziś już wypierana przez TLS, opracowany na bazie tej wersji.

Wersje protokołów TLS



- TLS 1.0 (SSL 3.1) – zdefiniowana w RFC 2246 w styczniu 1999, wprowadzono znaczące różnice w stosunku do SSL 3.0, częściowo złamany we wrześniu 2011.
- TLS 1.1 (SSL 3.2) – zdefiniowana w RFC 4346 w kwietniu 2006, zalecana przez *IETF The Internet Engineering Task Force* jako standard, coraz częściej używana, dodano zabezpieczenia przeciw atakom CBC (wektor IV).
- TLS 1.2 (SSL 3.3) – zdefiniowana w RFC 5246 w sierpniu 2008, wprowadzono nowe funkcje haszujące (SHA-256), obsługę AES, wersja ta została udoskonalona w marcu 2011 – nie będzie negocjować SSL 2.0.

Części składowe protokołu

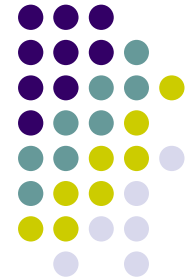
- Protokół SSL składa się z dwóch podprotokołów:
  - *SSL Record* – definiuje format przesyłanych pakietów danych,
  - zbiór trzech składowych do nawiązywania połączenia, renegocjowania i zgłaszania błędów:

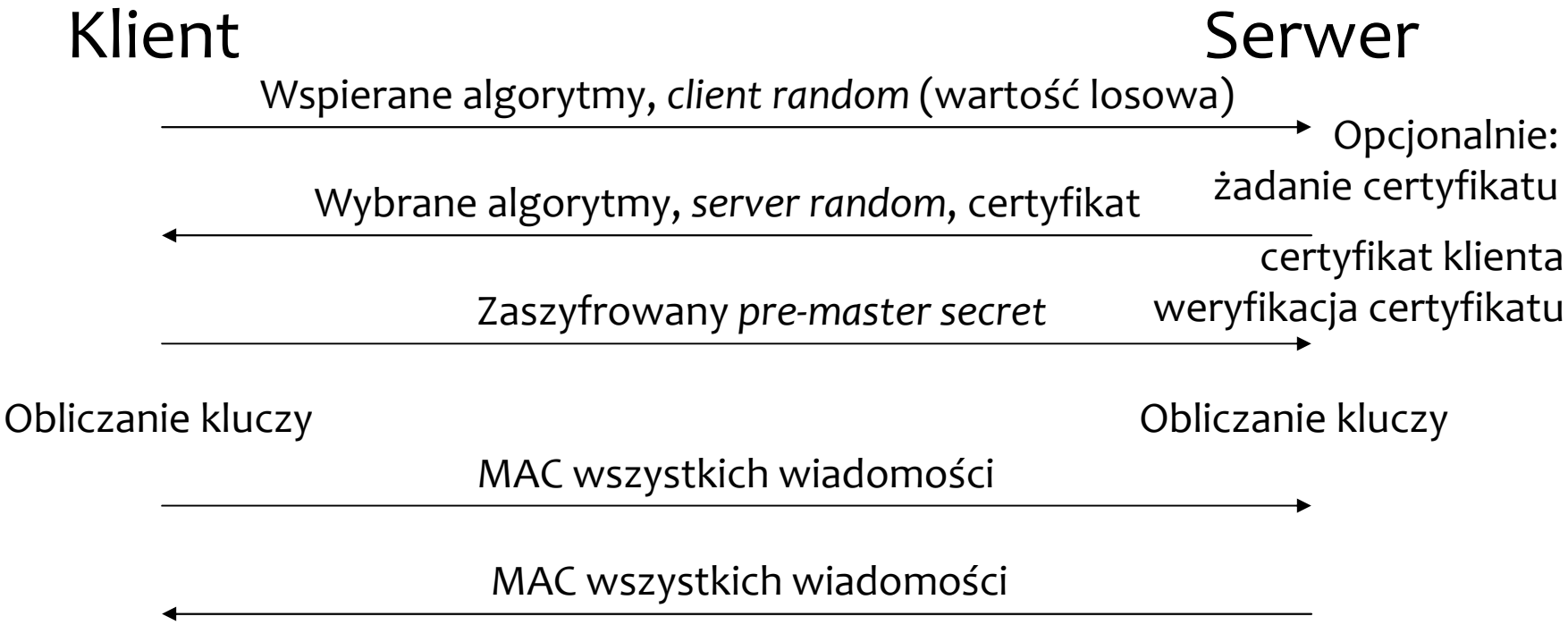
    SSL Handshake Protocol – ustala parametry bezpiecznego połączenia (algorytmy szyfrowania, uwierzytelniania i integralności danych),

    SSL Change Cipher Spec Protocol – protokół zmiany specyfikacji,

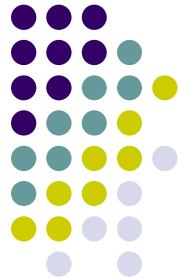
    SSL Alert Protocol – protokół zgłaszania błędów.

Szkic fazy handshake protokołu



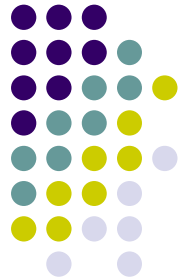


Składowe protokołu



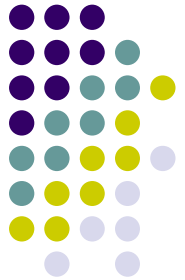
- Prześledzimy teraz wszystkie składowe protokołu, podprotokoły oraz wiadomości przesyłane w trakcie nawiązywania połączenia.

Record Layer



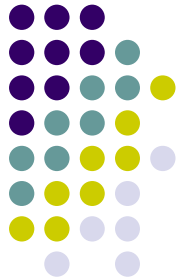
- Za nagłówkiem przesyłana jest wiadomość.
- Może być skompresowana (przed zaszyfrowaniem), wynegocjowanym algorytmem. Niestety protokół SSL/TLS nie przewiduje konkretnych algorytmów kompresji, zatem jest ona pomijana.
- Po „kompresji” obliczana jest wartość MAC (*Message Authentication Code*) i dodawana do pakietu.

Record Layer



- Obudowuje przesyłane wiadomości (pakiety danych, rekordy) dla następnych warstw (głównie TCP/IP).
- Każdy rekord może zawierać co najwyżej 2^{14} bajtów danych (16 384 bajtów, 16 kB), wiadomości zatem muszą być przycięte na odpowiednie fragmenty.
- W rekordzie mogą się znaleźć różne wiadomości, pod warunkiem, że są tego samego typu (wykorzystywane do przyspieszenia fazy handshake).
- Rekord zaczyna się nagłówkiem SSL zawierającym: wersję protokołu, długość danych w bajtach, typ zawartości (wiadomość, alert, dane handshake, itp.).

Record Layer



- Jeżeli używany jest symetryczny szyfr blokowy, wówczas dodawane są dane wyrównujące długość bloku, tak by długość wiadomości była wielokrotnością długości bloku.
- Za dodanymi informacjami, podawana jest ich długość (maksymalnie 255 bajtów).
- W przypadku braku szyfrowania (*NULL cipher*) lub szyfru strumieniowego dane wyrównujące są zbędne, przykładowo na etapie handshake – dane nie są szyfrowane.

Message Authentication Code

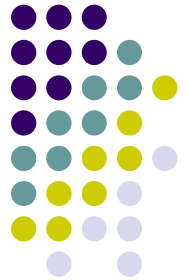
- MAC jest wyznaczany w następujący sposób, za pomocą funkcji *HMAC*

  $HMAC(K, text) = H(K \text{ XOR } opad \mid H((K \text{ XOR } ipad) \mid text))$,

  gdzie:

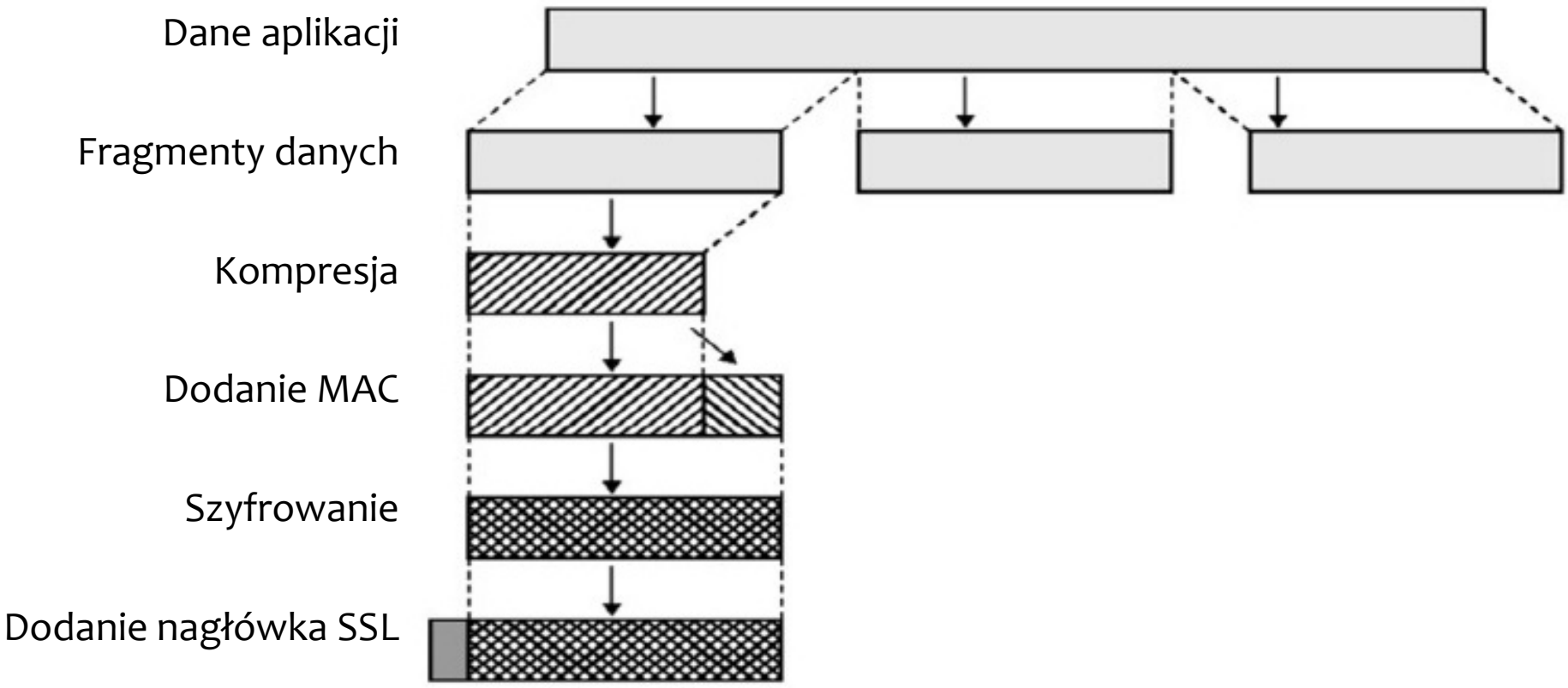
|	konkatenacja ciągów,
text	tekst jawny,
H	funkcja haszująca,
B	długość bloku funkcji *H* w bajtach,
K	klucz o długości co najwyżej *B* bajtów,
ipad	bajt 0x36 powtórzony *B* razy,
opad	bajt 0x5C powtórzony *B* razy.

Message Authentication Code



- W przypadku szyfrowania danych jako funkcję haszującą można używać MD5 lub SHA-1.
- Klucz jest ustalany podczas fazy hadshake, jest kluczem tajnym, co poprawia bezpieczeństwo np. w przypadku MD5.
- MAC jest stosowany zarówno do nagłówka jak i do danych w rekordzie, po zastosowaniu kompresji (pomijanej).
- MAC jest szyfrowany zgodnie z dalszą fazą protokołu.

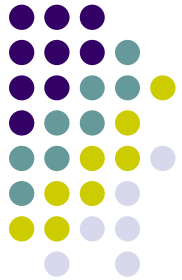
Record Layer



Przesyłanie wiadomości

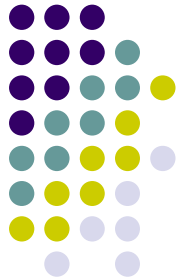
- Wiadomość jest podstawową jednostką w protokole SSL/TLS.
- Są cztery rodzaje wiadomości:
  - z fazy *Handshake* do nawiązania połączenia,
  - *Change cipher suite* do zmiany parametrów szyfrów,
  - *Alert* do zgłaszania błędów,
  - Wiadomości danych pomiędzy aplikacjami na końcach połączeń.

Faza handshake



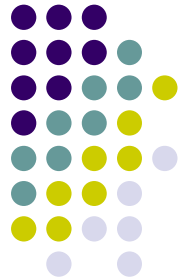
- Faza handshake służy do nawiązania połączenia oraz ustalenia parametrów szyfrowania danych.
- Wiadomości tej fazy zawierają tekst oraz długość przesyłanej wiadomości.
- Wiadomość może być jednego z poniższych typów: *hello request*, *client hello*, *server hello*, *certificate*, *server key exchange*, *certificate request*, *server hello done*, *certificate verify*, *client key exchange*, *finished*.

Hello Request



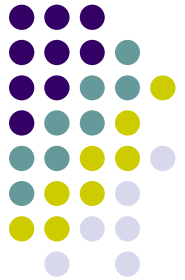
- Jest to pusta wiadomość, może być wysłana przez serwer do klienta w celu nawiązania połączenia.
- Klient może ją zignorować, jeżeli już negocjuje połączenie lub gdy nie chce renegocjować parametrów sesji.
- Klient może również opcjonalnie wysłać *no renegotiation alert*.
- Klient nie musi wysyłać takiej wiadomości, ma zastosowanie do renegocjowania kluczy przez serwer przy długoterminowych połączeniach.
- Klient wysyła *client hello*.

Client Hello



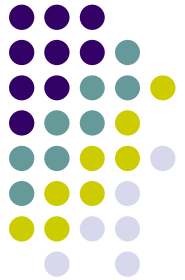
- Wysyłane gdy klient chce nawiązać połączenie z serwerem.
- Zawiera standardowy nagłówek rekordu, potem wersję klienta (najwyższa wersja protokołu obsługiwana przez klienta), potem losową liczbę (aktualny czas unix GMT oraz 28 bajtów z generatora pseudolosowego), następnie identyfikator sesji, następnie lista algorytmów obsługiwanych przez klienta (*cipher suites*), dotyczy szyfrowania, haszowania, wymiany klucza (opisane przez dwie 8 bitowe liczby), serwer będzie wybierał odpowiednie algorytmy, a jeżeli nie umie żadnego obsłużyć, to zgłasza błąd i zrywa połączenie.
- Dalej powinny być dwa bajty opisujące metodę kompresji, ale tylko NULL jest dozwolone (0x00).
- Następnie serwer odpowiada *server hello*.

Server Hello



- Odpowiedź serwera na *client hello*.
- Podobnie, zaczyna się od nagłówka, a potem jest wersja serwera (taka sama jak klienta jeżeli jest wspierana lub najwyższa jaką wspiera serwer), potem jest losowa liczba (analogicznie jak u klienta), identyfikator sesji (jeżeli taki sam jak klienta, to skraca się handshake i wykorzystuje wygenerowane i wcześniej zapisane klucze), można id pominąć wówczas od nowa wykonywany jest pełny handshake, potem są *cipher suites* akceptowane przez serwer.

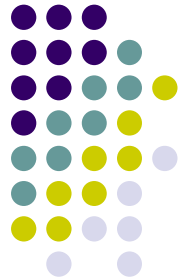
Cipher Suites



- Pełną listę *Cipher Suites* w protokole TLS można znaleźć na stronie

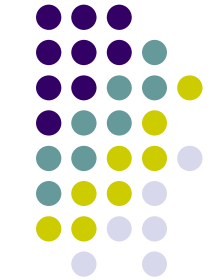
  http://www.iana.org/assignments/tls-parameters/tls-parameters.xml

Server Certificate, Client Certificate



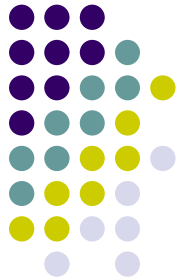
- *Server Certificate* wysyłany jest po *server hello*, jeżeli chcemy negocjować nie anonimowe połączenie, czyli prawie zawsze.
- Po nagłówku przesyłane są certyfikaty X.509, zaczynając od certyfikatu serwera.
- *Client Certificate* analogicznie, wysyłany na żądanie serwera przez klienta.

Server Key Exchange



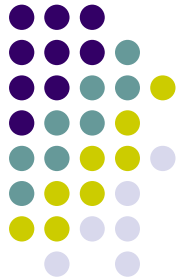
- Wiadomość wysyłana po *Server Certificate* jeżeli klient nie ma wystarczających danych do wymiany *pre-master secret* lub w przypadku anonimowych połączeń.
- Wykorzystywana do anonimowego protokołu Diffie-Hellmana (*DH*), efemerycznego DH (*Ephemeral DH*) oraz efemerycznego RSA (*Ephemeral RSA*) służących do wymiany klucza symetrycznego.

Certificate Request



- Serwer może zażądać przesłanie certyfikatu przez klienta w celu uwierzytelnienia.
- Po nagłówku wysyłana jest lista żądanych typów certyfikatów (możliwe są *rsa sign*, *dsa sign*, *rsa fixed dh*, *dss fixed dh*), a następnie lista akceptowanych urzędów certyfikujących.

Server Hello Done

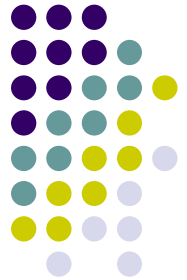


- Wiadomość wysyłana przez serwer do zakomunikowania zakończenia fazy handshake, czyli serwer nie będzie wysyłał: *server hello*, *server certificate*, *certificate request*.
- Zawiera wyłącznie nagłówek rekordu.

Client Key Exchange

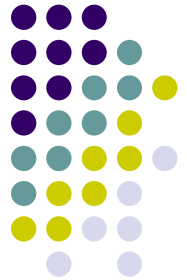
- Wysyłana po *Client Certificate* lub po *Server hello done*.
- Występuje w wersji RSA oraz Diffie-Hellman w zależności od metody wyznaczania klucza sesyjnego.
- W przypadku RSA, po nagłówku wysyłany jest zakodowany zgodnie z PKCS #1 *pre-master secret* (zaszyfrowany kluczem publicznym serwera z certyfikatu), nagłówek nie jest szyfrowany.

Client Key Exchange



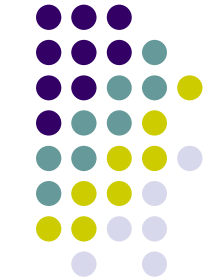
- Wartość *pre-master secret* składa się z najwyższego wspieranego przez klienta numeru wersji oraz 48 losowych bajtów z generatora. Jest to wkład klienta do *master secret* z którego powstaną klucze sesyjne.
- Wersja jest potrzebna do ochrony przed atakiem man-in-the-middle.
- Kodowanie PKCS #1 służy do ustalenia długości *pre-master secret* odpowiadającej długości klucza RSA.

Client Key Exchange



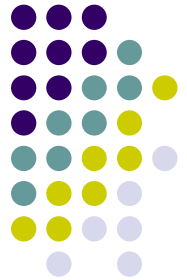
- W przypadku DH, wiadomość po nagłówku zawiera publiczne parametry protokołu DH, co umożliwia klientowi i serwerowi obliczenie tej samej wartości *pre-master secret*.
- Jeżeli wymagane było uwierzytelnienie klienta, to w jego certyfikacie mogą być zapisane publiczne wartości potrzebne do protokołu DH, wówczas ta wiadomość składa się tylko z nagłówka.

Certificate Verify



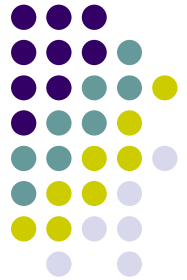
- Weryfikacja wiadomości od klienta, umożliwia sprawdzenie czy rzeczywiście klient jest posiadaczem przesłanego certyfikatu (musi zatem mieć odpowiedni klucz prywatny).
- Wysyłana jest po *Client Certificate*, zawiera podpisany skrót wszystkich dotychczasowych wiadomości fazy handshake (wysłanych i odebranych). Podpis ten jest weryfikowany kluczem z certyfikatu klienta.
- W przypadku podpisu RSA podpisywana jest konkatenacja skrótów MD5 i SHA. W przypadku DSA podpisywany jest skrót SHA-1.

Finished



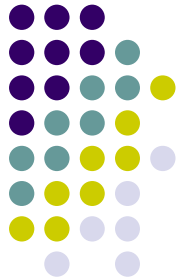
- Wysyłana przez klienta i przez serwer po wiadomości *change cipher spec*, w celu weryfikacji czy faza handshake nie jest modyfikowana przez niepowołane osoby.
- Wykorzystywana jest do tego PRF (*Pseudo Random Function*) oraz *master secret*. Dodawane są również słowa „client” i „server” by wartości po obu stronach były różne (utrudnia atak przez powtórzenie w przypadku man-in-the-middle).

Change Cipher Spec



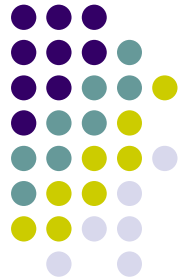
- Wiadomość wysyłana by powiadomić drugą stronę, że parametry wynegocjowane jak dotąd będą stosowane już od następnej wiadomości (zmiana parametrów szyfrowania).
- Wysyłana na zakończenie fazy handshake ale przed danymi aplikacji.
- Wysyłana jednokierunkowo, zatem musi być przesłana dwa razy z każdego końca.

Alert



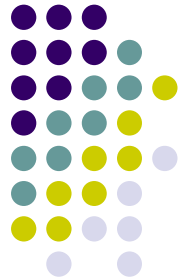
- Wiadomość o błędzie, może być wysłana w dowolnym momencie, przez klienta lub serwer.
- Zamyka połączenie i zgłasza wystąpienie błędu.
- Zawiera poziom błędu (ostrzeżenie lub błąd) oraz opis, dwie liczby 8 bitowe.

Application Data



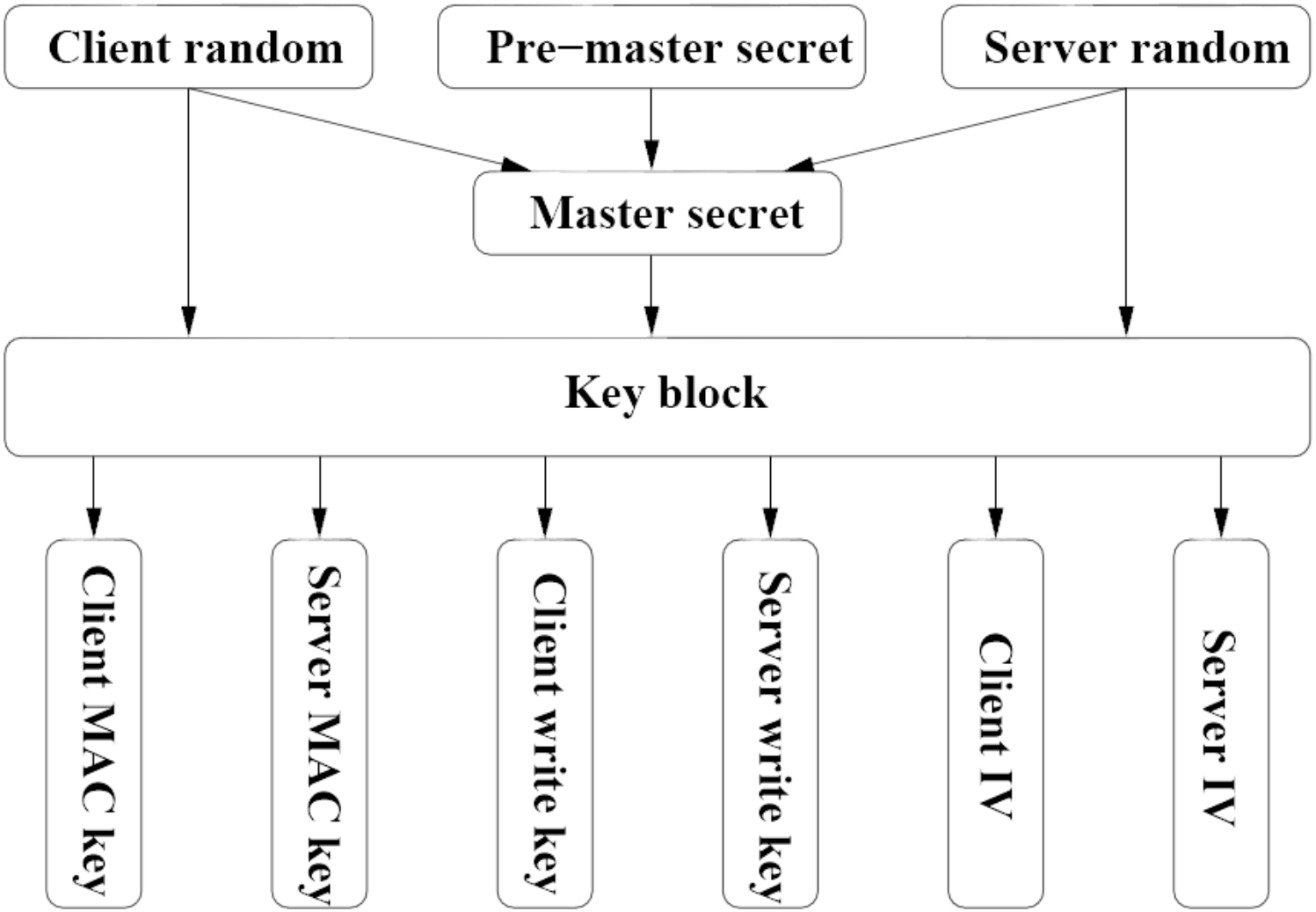
- Ostatni rodzaj wiadomości, przenosi dane aplikacji pomiędzy końcami nawiązanego połączenia.
- Nie posiada nagłówka, tylko dane zapisane w postaci rekordu.

Generowanie kluczy



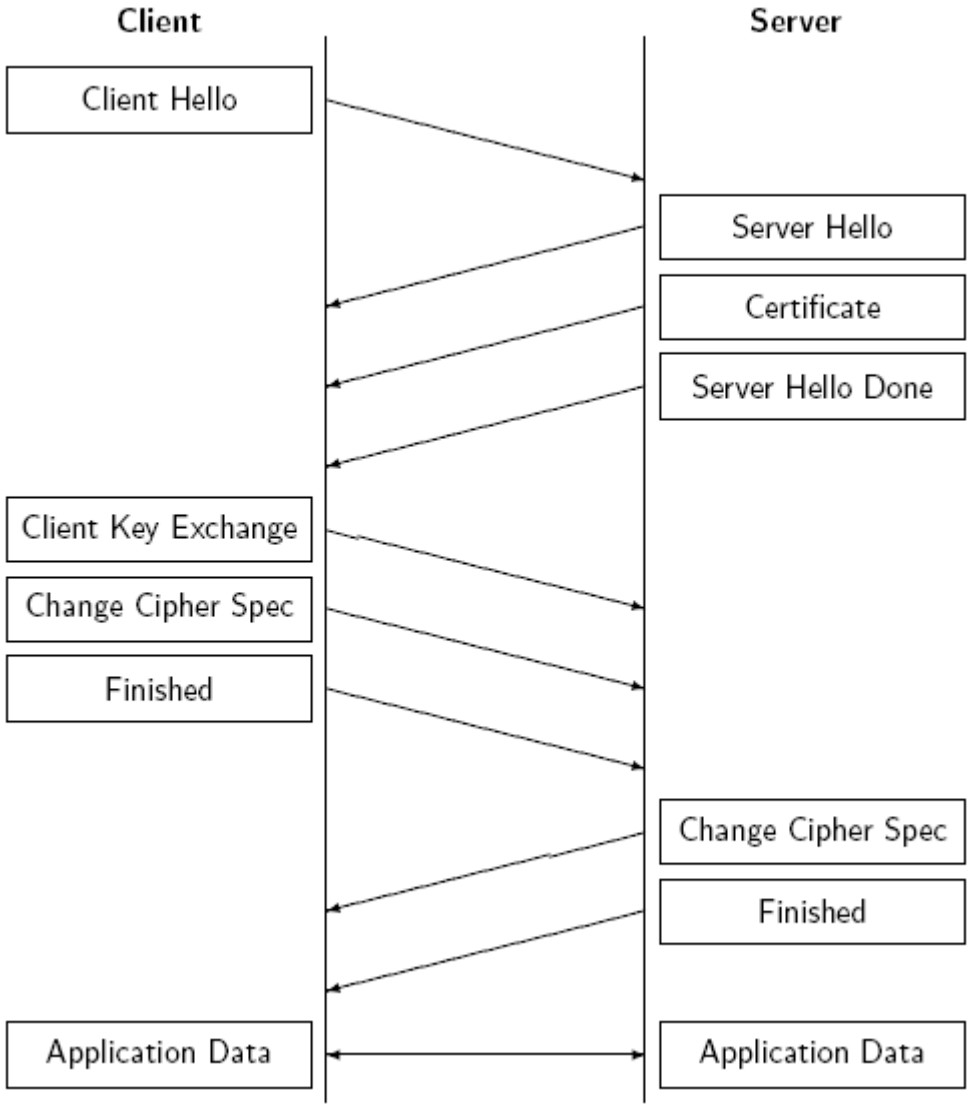
- Tajne wartości wykorzystywane są podczas szyfrowania, weryfikacji jako klucze lub wektory początkowe.
- Generowane są za pomocą algorytmu na podstawie losowych wartości przesłanych w wiadomości *client hello* i *server hello* oraz wartości *pre-master secret* z wiadomości *client key exchange*.

Generowanie kluczy



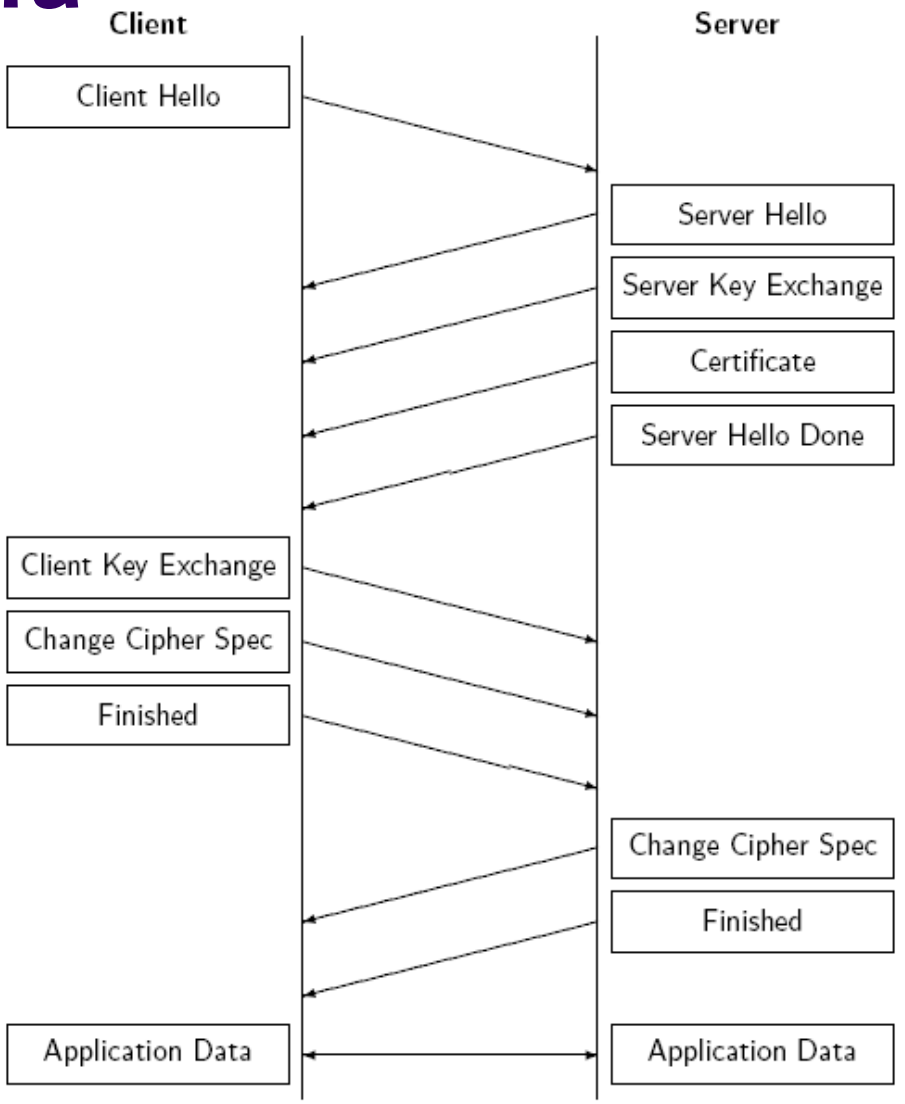
Przebieg protokołu

- Handshake
  z wykorzystaniem
  RSA



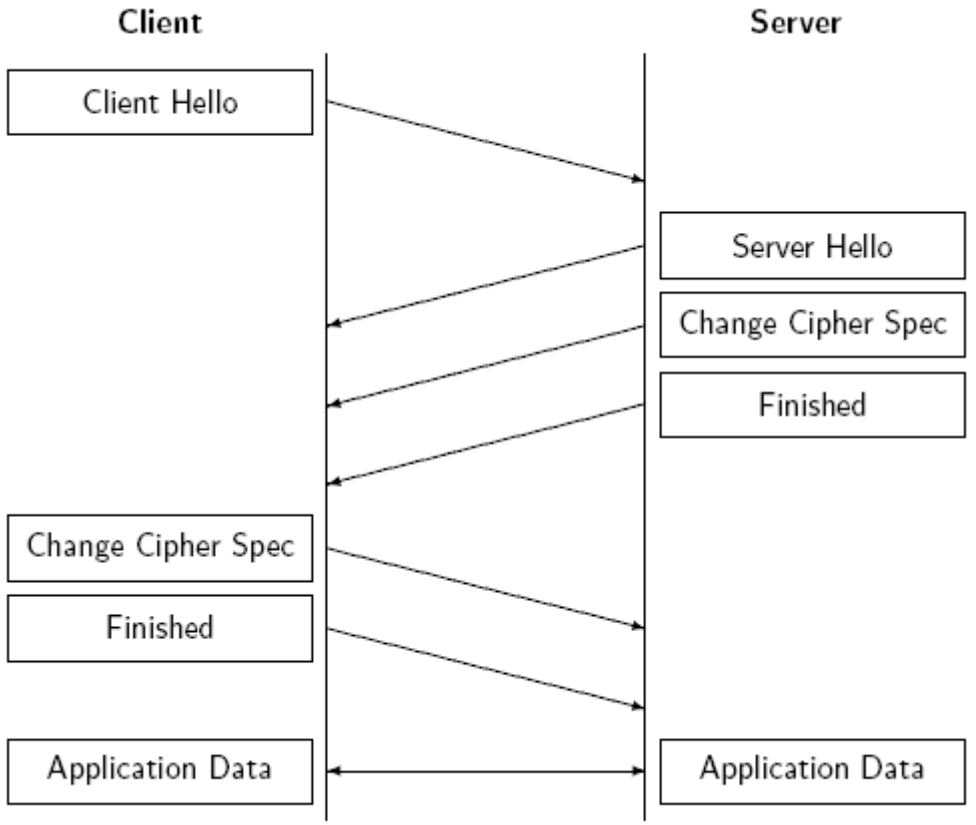
Przebieg protokołu

- Efemeryczny handshake (wykorzystuje RSA lub DH)

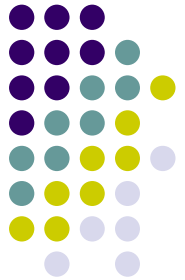


Przebieg protokołu

- Wznawianie połączenia



Bezpieczeństwo

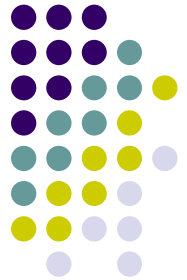


- *Phishing* – logowanie na kopię strony, certyfikat wystawiony na innego wystawcę, np. o podobnej nazwie.
- *Root Certificate Poisoning* – wstawianie do przeglądarki nowych centrów certyfikacji i wystawianie certyfikatów poświadczonych przez te centra.

Bezpieczeństwo

- 23 września 2011 Thai Duong i Juliano Rizzo przedstawili nowy atak na TLS i program BEAST (*Browser Exploit Against SSL/TLS*), wykorzystujący applet w Javie. Wykorzystuje on tzw. lukę CBC w TLS 1.0 odkrytą w 2002 roku przez Phillipa Rogawaya.

Bezpieczeństwo



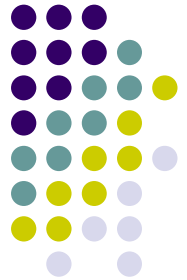
- Mozilla już dodała łatę do Firefoxa naprawiającą protokół, również Google Chrome został poprawiony. Niestety niektóre webserwery (wykorzystujące stare biblioteki) mogą nie ustanawiać połączenia. Łatwą poprawą jest wyłączenie wszystkich szyfrów CBC z listy akceptowanych szyfrów. Microsoft wypuściło Security Bulletin MS12-006 w styczniu2012 do TSL 1.1 poprawiającą bezpieczeństwo w webserwerach i przeglądarkach Microsoftu.

Bezpieczeństwo

- BEAST umożliwia m.in. przechwycenie danych z zaszyfrowanych plików cookie (np. loginów i haseł do stron WWW), ale tylko tych zaszyfrowanych AES. Wykorzystywanie np. RC4 nie jest podatne na atak.
- Wersja TLS 1.1 i 1.2 nie jest podatna na atak, ale jeszcze nie jest powszechnie używana w przeglądarkach.
- Przykład wykorzystania BEAST do ataku na PayPal:

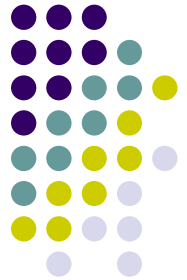
  http://www.youtube.com/watch?v=BTqAIDVUvrU

Bezpieczeństwo



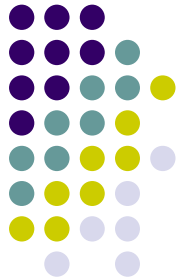
- Użytkownicy Windows 7, Windows 8 oraz Windows Server 2008 R2 mogą już używać TLS 1.1 oraz 1.2, pod warunkiem, że druga strona również będzie używać poprawionej wersji, jeżeli nie zostanie użyta wersja 1.0 protokołu TLS.
- W październiku 2013, około 69.5% (+4.4%) stron internetowych używało nadal podatnych na atak BEAST wersji protokołu

Bezpieczeństwo



- W roku 2013 pojawiło się wiele nowych ataków na protokół TLS:
  - Odmiany BEAST, ataki CRIME i BREACH
  - Padding attack
  - Atak na RC4 (BEAST wykorzystywał jedynie AES)

Następny wykład
Praca własna



- Bezpieczeństwo warstwy sieciowej
- Protokół IPSec