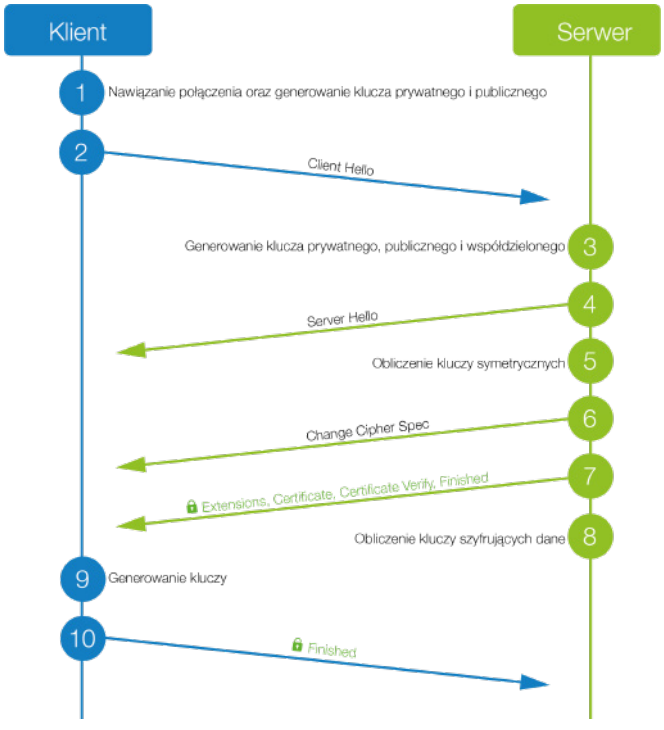
Co zabezpiecza HTTPS, czyli o protokole TLS 1.3

Protokół HTTPS, a dokładniej HTTP over TLS, jest rozszerzeniem protokołu HTTP o funkcjonalność szyfrowania przesyłanych danych wraz z możliwością uwierzytelnienia klienta i serwera. Obie te funkcje możliwe są dzięki zastosowaniu protokołu TLS (Transport Layer Security). W tym artykule przedstawiony zostanie krok po kroku proces nawiązania szyfrowanego połączenia, który posłuży do wymiany szyfrowanych danych w HTTPS.

TLS 1.3

Protokół TLS 1.3 miał swoją premierę w sierpniu 2018 i powstał jako następca TLS 1.2, który do tej pory jest szeroko używany, a dobrze skonfigurowany dalej uznawany jest za bezpieczny. Główne zmiany w nowszej wersji obejmują redukcję wymaganych połączeń między klientem a serwerem w procesie negocjacji bezpiecznego połączenia, usunięcie algorytmów, w których wykryto podatności bezpieczeństwa, usunięcie możliwości kompresji danych, dodanie nowych krzywych eliptycznych czy wprowadzenie mechanizmu 0-RTT przyspieszającego wznowienie połączenia. Dla łatwiejszego zrozumienia działania protokołu TLS 1.3 artykuł ten nie będzie poruszał wszystkich nowości ani wszystkich możliwości protokołu, do których należy także sprawdzanie poprawności certyfikatów czy uwierzytelnianie certyfikatem.

Na Rysunku 1 przedstawione zostały główne kroki procesu negocjacji bezpiecznego połączenia TLS i to na nim zostanie oparta dalsza część artykułu. Zielona kłódka przy nazwie przesyłanego pakietu oznacza, iż jest on szyfrowany.



Rysunek 1. Proces negocjacji bezpiecznego połączenia

KROK 1. NAWIĄZYWANIE POŁĄCZENIA TCP I GENEROWANIE KLUCZY

Pierwszym krokiem, jaki należy wykonać, jest nawiązanie zwykłego połączenia TCP[1] oraz wybór parametrów szyfrowania i algorytmów podpisywania certyfikatów. Następnie generowane są pary klucza prywatnego i klucza publicznego. Klucz prywatny jest pseudolosową, bardzo dużą liczbą nie przekraczającą wartości 256-1[2]. Klucz publiczny generowany jest z klucza prywatnego za pomocą krzywych eliptycznych poprzez wymnożenie punktu bazowego[3] [4] leżącego na krzywej eliptycznej z kluczem prywatnym jako skalar. Wynikiem tej operacji jest pewien punkt na krzywej eliptycznej i jest to właśnie klucz publiczny, który zapisuje się jako konkatenację współrzędnych x i y. Cechą krzywych eliptycznych jest to, iż znając punkt początkowy (punkt bazowy), praktycznie niemożliwe jest obliczenie[5], przez jaką wartość (klucz prywatny) został on pomnożony, uzyskując dany punkt na krzywej (klucz publiczny). To zagadnienie to tzw. problem logarytmu dyskretnego.

KROK 2. CLIENT HELLO

Po nawiązaniu połączenia i wygenerowaniu kluczy do serwera wysyłany jest pakiet „Client Hello", który zawiera m.in. preferowane algorytmy szyfrowania wraz z funkcjami mieszającymi, wygenerowane klucze publiczne, listę wspieranych krzywych eliptycznych czy listę algorytmów weryfikacji certyfikatu. Lista dostępnych algorytmów szyfrowania względem poprzedniej wersji TLS została skrócona o te algorytmy, w których wykryto podatności bezpieczeństwa. W Listingu 1 przedstawiony jest przykładowy pakiet dla protokołu TLS 1.3 wraz z opisem poszczególnych pól. Obok danej wartości pola w nawiasach zamieszczona jest wartość w kodzie szesnastkowym.

Listing 1. Przykładowy pakiet Client Hello

```
1. Content Type: Handshake (0x16)
2. Version: TLS 1.0 (0x0301)
3. Length: 138
4.   Handshake Type: Client Hello (0x01)
5.   Length: 134
6.   Version: TLS 1.2 (0x0303)
```

1. Istnieje także możliwość użycia protokołu UDP, wtedy mówimy o DTLS, ale nie jest on wykorzystywany w HTTPS.
2. Wartość klucza prywatnego ma także inne obostrzenia w zależności od użytej krzywej eliptycznej.
3. Punkt bazowy zależny jest od użytej krzywej eliptycznej.
4. Często zamiast określenia punktu bazowego można spotkać się z terminem „generator".
5. Problem z obliczeniem takiej wartości wynika z czasochłonności takiego procesu dla dużego skalaru.