Vysoké učení technické v Brně
Fakulta informačních technologií

1. projekt do předmětu KRY
Tunel pro bezpečné zasílání zpráv

Martin Janoušek, 5. dubna 2017

1 Úvod

Cílem projektu bylo vytvoření tunelu pro bezpečné zasílání zpráv mezi dvěma účastníky – klientem a serverem. Při realizaci měl být využit algoritmus Diffie-Hellman pro generování klíče, dále identifikace pomocí Feige-Fiat-Shamirova schématu, symetrické šifrování AES a hashovací funkce SHA256.

2 Implementace

2.1 Komunikační protokol

Komunikační protokol se skládá ze dvou částí:

- Ustanovení klíče pomocí algoritmu DH,
- Šifrovaná komunikace pomocí šifry AES.

2.1.1 Ustanovení klíče

Pro ustanovení klíče je nutné odeslat večejný klíč clienta `pub_c` na server a poté odeslat veřejný klíč serveru `pub_s` klientovi. V projektu je tedy výměna implementována v tomto pořadí.

2.1.2 Komunikace

Následná šifrovaná komunikace je vždy inicializována klientem. V projektu je komkrétně implementováno odesální 5 náhodných zpráv vygenerovaných pomocí funkce `generateString()` ze strany klienta. Na každou zprávu server odpoví jejím hashem, který je v zašifrované formě odeslán zpět na klienta. Klient po přijetí hashe od serveru porovná, zda se hash shoduje s hashem zprávy, kterou odeslal. Počet odeslaných zpráv je nastaven konstantou `ITERATION` a může být v případě potřeby při hodnocení projektu navýšen.

Výměna jedné zprávy lze tedy zjednodušené popsat následující posloupností kroků:

1. CLIENT: Vygenerování zprávy `m`.
2. CLIENT: Zašifrování zprávy `m` na šifrovaný text `c`.
3. CLIENT: Odeslání šifrovaného textu `c`.
4. SERVER: Přijetí šifrovaného textu `c`.
5. SERVER: Dešifrovaní zprávy `m` ze šifrovaného textu `c`.
6. SERVER: Spočítání hashe `hs` zprávy `m`.
7. SERVER: Zašifrování hashe `hs` na šifrovaný text `chs`.
8. SERVER: Odeslání `chs`.
9. CLIENT: Přijetí `chs`.
10. CLIENT: Dešifrování `chs` a získání `hs`.
11. CLIENT: Spošítání hashe `hm` z originálu zprávy `m`.
12. CLIENT: Porovnání `hs` a `hm`.

2.2 Diffie-Hellman

Pro ustanovení klíče je použit algoritmus Diffie-Helman (dále DH). Při jeho implementaci bylo využito knihovny *OpenSSL*. V algoritmu DH je nejprve nutné ustanovit parametry `g` a `p`. Tyto parametry mohou být veřejné a musejí být totožné na serveru i klientu. Vygenerování probíhá ve funkci `getDH2048`, která byla přímo vygenerována pomocí unixového příkazu: `openssl dhparam -C 2048`. Tyto parametry musejí být předpočítány dopředu, protože jejich generování je časově náročné. Po ustanovení parametrů si klient/server vegeneruje veřejný a soukromý klíč pomocí funkce `DH_generate_key` a veřejný klíč odešle druhé straně (serveru/klientu). Po získání veřejného klíče protější strany si každá strana vygeneruje sdílené tajmeství (klíč) pomocí funkce `DH_compute_key`. Celý tento proces je implementován ve funkci `init_DH`.

2.3 Redukce klíče

Klíč spočítaný v předchozím kroce je nutné redukovat na 256 bitů. Dále je nutné získat inicializační vektor dlouhý 128 bitů. Kokrétně je v projektu tedy redukovaný klíč prvními 256 bity klíče vygenerovaného v předchozím kroce a inicializační vektor je dalšími 128 bity, které následují.

2.4 Feige-Fiat-Shamirovo identifikační schéma

Nebylo implementováno z časových důvodů.

2.5 AES

Šifra AES je použita v režimu CBC s 256 bitovými bloky dat. Šifrování probíhá ve funkci `encrypt` a je při něm použito opět knihovny *OpenSSL*. Kokrétně jsou použity funkce `EVP_EncryptInit_ex`, `EVP_EncryptUpdate` a `EVP_EncryptFinal_ex` pro inicializaci šifry, šifrování a dokončení šifrování. Dešifrování probíhá pomocí stejných funkcí jen s prefixem `EVP_Decrypt`.

2.6 Úskalí v zabezpečení komunikace zadaným způsobem

Vzhledem k chybějícímu Feige-Fiat-Shamirovu identifikačnímu schématu je zřejmé, že implementace je náchylná k útoku Man-in-the-Middle.