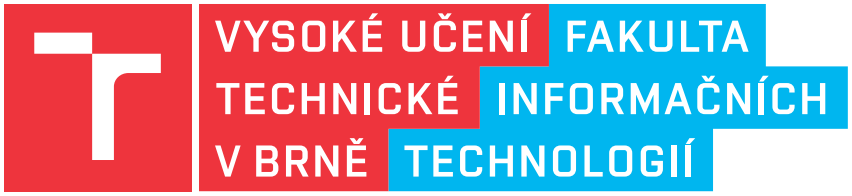

Implementace hybrydního šifrování

KRY - Kryptografie

27. března 2023

Bc. Jakub Komárek (xkomar33)

1 Úvod

Tento projektu se věnuje hybridní šifrované komunikaci, konkrétně s využitím algoritmů AES 128 bit, RSA 2048 a MD5 hash. Cílem projektu je implementace aplikace klient-server s bezpečným předáváním zpráv pomocí hybridního šifrování.

2 Implementace

Zdrojový kód je členěn na soubory podle jejich funkce následujícím způsobem:

- kry.py - main soubor, parsuje argumenty, volá hlavní funkce programu
- keyGen.py - funkce na vytváření klíčů
- methods.py - implementace kryptografických funkcí
- client.py - implementace síťového klienta
- server.py - implementace síťového serveru
- cert/ - složka s klíči - vytvoří se a naplní se při spuštění programu

3 Použité kryptografické algoritmy a jejich klíče

Při implementaci jsem se moc neodchýlil od zadání, implementaci jsem provedl 1:1.

3.1 Seznam klíčů

- clientKey - Soukromý klíč klienta. Nikdy se nepřenáší a vlastního každý klient. Je užit pro podepisování odesílaných zpráv
- clientKeyPub - Veřejný klíč každého klienta. Může ho vlastnit kdokoliv. Posílá se před začátkem přenosu cypher textu.
- serverKey - Soukromý klíč serveru. Nikdy se nepřenáší a vlastního server. Je užit na rozšifrování klíče relace.
- serverKeyPub - Veřejný klíč serveru. Může ho vlastnit kdokoliv. Používá se pro šifrování klíče relace zprávy určené pro daný server. Přenáší se na začátku komunikace nebo může být externě dodán CA.

3.2 AES - symetrické šifrování

V projektu je symetrické šifrování užito pouze pro šifrování/dešifrování celé zprávy s podpisem. Šifrování/dešifrování je provedeno za pomocí jednoho sdíleného klíče (klíč relace). Tento klíč je zašifrován asymetricky pomocí RSA veřejným klíčem příjemce (serverKeyPub) a takto zabezpečený klíč je přiložen na konec zprávy.

Klíč je vytvořen zcela náhodně pro každou odeslanou zprávu za použití funkce get_random_bytes z knihovny PyCryptodome. Tato funkce je postavena na funkci urandom, která je považována za kryptograficky bezpečnou.

Vytvoření AES klíče probíhá u každé vytvořené zprávy. Délka klíče je 128 bitů, což je považované za standardní u tohoto algoritmu a poměrně bezpečné. Menší klíče by měly zásadní vliv na bezpečnost, kritická velikost klíče se nachází okolo 54 bitů. Pro použití vyžadující větší bezpečnost bych raději volil 256 bitovou variantu algoritmu.

3.3 RSA - asymetrické šifrování

V projektu je toto šifrování využito 2x. Prvně je použito pro zašifrování klíče relace (viz. kapitola AES). Klíč relace je šifrován veřejným klíčem příjemce (serverKeyPub). Distribuci tohoto klíče by měla zajistit certifikační autorita, jinak si nebudeme jistí, zda zprávy neposíláme útočníkovi. Soukromý klíč příjemce (serverKey) má pouze a jenom server.

Druhé použití asymetrického šifrování je při podpisu zprávy. Z každé zprávy určené k odeslání je spočítán kontrolní součet, který je následně zarovnán a podepsán soukromým klíčem odesílatele (clienKey). Tento klíč vlastní pouze a jenom klient. Vzhledem k faktu, že se klient nijak neautentizuje musí opět veřejný klíč (client-KeyPub) rozdistribuovat certifikační autorita, jinak nelze s jistotou říci s kým server komunikuje.

Záleží tedy na účelu použití - pokud nepotřebujeme splnit bezpečností cíl autentizace, můžeme veřejné klíče distribuovat při komunikaci - tento způsob byl použit v projektu. Veřejné klíče si mezi sebou vymění klient se serverem v nešifrované komunikaci - možný útok man-in-the-middle.

Každý pár klíčů je vytvořen pomocí funkce RSA.generate z knihovny PyCryptodome a to pouze při prvním spuštění aplikace. Délka klíče je podle zadání 2048 bitů, což je požadované za nutný základ, menší klíče by měli velký dopad na bezpečnost. Pro použití vyžadující větší bezpečnost bych raději volil 4096 bitovou variantu algoritmu případně 3072 bitovou. Větší délka klíče bude mít také za následek možnosti přenášet větší zprávy. U 2048 jsme omezeni na velikost 214 bajtů.

3.3.1 Padding

Můj návrh paddingu vyšel z principu, že známe velikosti doručovaných zpráv. Rozhodl jsem se tedy zbytek zarovnaného prostoru vyplnit náhodnými znaky - osolit text určený k šifrování. Tudíž dvě stejné zprávy budou pokaždé mít jiný cypher text.

Výhodou tohoto řešení je jednoduchost implementace. Nevýhodou je již zmíněná nemožnost zjištění předělu padingu. Pokud bych tuto apriorní informaci (délka textu) neznal musel bych použít některý jiný způsob. K tomuto datu se jeví jako nejlepší alternativa implementace podle standardu RSAES-OAEP.

4 Formát zprávy

Každá posílaná část zprávy je zakódována do hexadecimální podoby. Tyto hexadecimální znaky jsou poté ještě jednou překódovány do binární podoby pomocí kodéru (utf-8). Tento přístup byl zvolen k zamezení konfliktů - jednotlivé části jsou odděleny znakem středníku, konec zprávy je poznačen escape sekvencí /004. Konflikt tedy nemůže vzniknout. Nevýhoda tohoto přístupu spočívá v delším přenášeném cypher textu. Optimalizovanější varianty by šlo dosáhnout použitím pokročilejším užitím kódovacích technik.

5 Chybné doručení a integrita

V případě chybného doručení zprávy se zpráva posílá celá znovu, není však přešifrována (posílá se ta samá zpráva). Po 5 neúspěšných pokusech se zpráva prohlásí za nedoručitelnou. Tato implementace má slabinu v podobě možné chybové analýzy - útočník ví jestli zpráva byla správně přijata nebo ne. Pro větší bezpečnost je vhodné tuto okolnost klientovy neoznamovat a posílat pouze ack zprávu v případě přijaté zprávy (nevzdělovat, zda-li byla správně zpracována).

Integritu paketů (náhodné chyby na přepravní vrstvě) zajišťuje TCP protokol.

6 Bezpečnostní cíle a audit

- Integrita - Každá zpráva je podepsána, tudíž je poznat jakákoliv modifikace při přepravě zprávy. Modifikace části s AES klíčem bude mít za následek chybné rozšifrování, což opět bude detekovatelné.
- Důvěrnost - Přenášený text je zašifrovaný, tudíž nelze bez znalosti klíče zjistit její obsah, podpis je díky paddingu a náhodnému klíči vždy jiný vzhledem ke stejnému obsahu zprávy. Tudíž nelze ani detekovat stejné zprávy. Jediné co by šlo o zprávě zjistit je její délka. Pokud tuto informaci chceme utajit, bylo by nutné zprávu zarovnat na nějakou konstantní velikost bloku.
- Nepopiratelnost - Díky podpisu zprávy můžeme zaručit, že zprávu poslal daný vlastník soukromého klíče. Nikdo jiný daný podpis není schopný vytvořit.
- Autentizace - Pokud by oba veřejné klíče byly rozdistribuovány pomocí certifikační autority - tudíž by bylo možné zkontrolovat totožnost obou stran, poté by byl tento bezpečností cíl splněn. S touto implementací však nelze ověřit totožnost komunikující protistrany.

  Alternativou by bylo rozdistribuování pouze veřejného klíče serveru pomocí CA. Klient by se poté přes zabezpečený kanál ověřil serveru například pomocí hesla.

7 Zhodnocení

Podle mého dojmu byl projekt vypracován v plném rozsahu.