Hybridní šifrování

V tomto projektu se seznámíte s hybridní šifrovanou komunikací, konkrétně s využitím algoritmů AES 128 bit, RSA 2048 a MD5 hash. Vašim projektem bude naimplementovat aplikaci klient-server s bezpečným předáváním zpráv pomocí hybridního šifrování.

Teorie:
Hybridní šifrování je kombinace dvou různých typů šifrování pro zajištění bezpečného přenosu dat přes nezabezpečené kanály, jako je například internet. Tento typ šifrování využívá silné stránky obou metod: symetrického a asymetrického šifrování.

Symetrické šifrování je rychlé a efektivní, ale vyžaduje sdílený tajný klíč mezi odesílatelem a příjemcem. Na druhou stranu **asymetrické šifrování** používá veřejný a soukromý klíč pro šifrování a dešifrování zpráv. Tato metoda je pomalejší, ale umožňuje komunikaci mezi stranami, které si navzájem nedůvěřují.

V hybridním šifrování se obvykle používá asymetrické šifrování pro bezpečnou výměnu tajného klíče pro symetrické šifrování neboli ustanovení klíče relace, který se následně používá pro samotné šifrování dat. To umožňuje bezpečnou výměnu informací mezi odesílatelem a příjemcem bez potřeby sdílet tajný klíč.

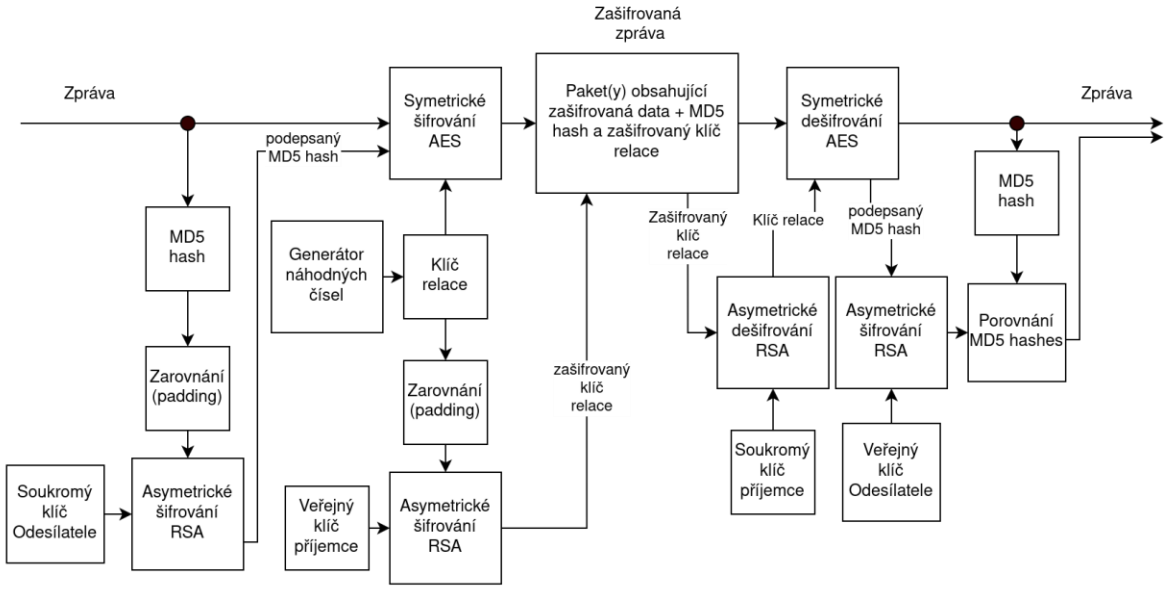
Využívané algoritmy:
RSA (Rivest–Shamir–Adleman) je asymetrický šifrovací algoritmus, který se často používá pro hybridní šifrování. RSA používá veřejný a soukromý klíč pro šifrování a dešifrování zpráv. Veřejný klíč se používá pro šifrování zprávy, kterou může dešifrovat pouze příjemce, který má přístup ke svému soukromému klíči.

AES (Advanced Encryption Standard) je symetrický šifrovací algoritmus, který se používá pro šifrování dat. Z důvodu rychlosti oproti asymetrické kryptografii se používá na šifrování většího množství dat než asymetrické šifrovací algoritmy. AES používá stejný klíč pro šifrování a dešifrování zprávy.

MD5 (Message-Digest Algorithm 5) vytváří 128bitový hash, což znamená, že výstupní hodnota je 128-bitové číslo, které reprezentuje jedinečný otisk vstupního textu. Hashovací funkce MD5 byla používána v mnoha aplikacích pro ověřování integrity dat, jako například v operačních systémech, databázích, webových prohlížečích a dalších aplikacích.

Architektura klient-server bude umožňovat zasílání zpráv zašifrovaných pomocí symetrické šifry AES 128bit a RSA 2048 bitů. Po spuštění aplikací se naváže spojení mezi klientem a serverem. Program bude podporovat vkládání zpráv v nekonečné smyčce. Každá zpráva se bude skládat z X znaků, které bude program načítat ze standartního vstupu ukončené stisknutím klávesy **enter**. Každá zpráva bude následně zašifrována pomocí algoritmu AES a klíč relace bude zašifrován pomocí algoritmu RSA.

Schéma odesílání zprávy



Nejprve je potřeba vytvořit kontrolní součet zadané zprávy pomocí algoritmu MD5, který se následně podepíše pomocí soukromého klíče odesílatele. Podepsaný kontrolní součet se připojí ke zprávě pro zajištění integrity zprávy. Pro symetrické šifrování zprávy a kontrolního součtu bude využit algoritmus AES (není potřeba implementovat vlastní generátor náhodných čísel). Klíč můžete získat například z /dev/random, v případně nutnosti /dev/urandom nebo jej vygenerovat pomocí existující knihovny. Pro asymetrické šifrování (RSA) můžete využít již existující řešení pro generování klíčů i pro šifrování/dešifrování. Vygenerované klíče nebo certifikáty pro asymetrické šifrování budou uloženy ve složce ”cert”.

Vašim úkolem bude vytvořit klíče pro symetrické a asymetrické šifrování. Následně pro vloženou zprávu ze standartního vstupu vypočítat kontrolní součet, který bude následně zarovnán a zašifrován pomocí asymetrické kryptografie. V dalším kroku bude kontrolní součet přiložen ke zprávě a celou zprávu zašifrovat pomocí symetrické kryptografie. Poslední část přenášených dat je klíč relace pro symetrické šifrování. Tento klíč je nejprve potřeba rozšířit na správný počet bitů a následně jej zašifrovat pomocí asymetrické kryptografie. Zašifrovaný klíč se připojí k zašifrované zprávě a je též odeslán.
Příjem zprávy bude fungovat následovně. Nejprve je potřeba rozdělit přijatá data na zašifrovaný klíč a zašifrovaná data. Následně je potřeba dešifrovat zašifrovaný klíč pomocí soukromého klíče příjemce. Pomocí získaného klíče lze dešifrovat přijatou zašifrovanou zprávu. Po dešifrování zprávy je potřeba provést její kontrolu integrity pomocí algoritmu MD5. Zprávu bude potřeba nejprve dešifrovat a zkontrolovat, zda se jedná o nezměněnou zprávu, také je potřeba bezpečně oznámit odesílateli, že zpráva byla správně přijata. Případně požádat odesílatele o znovu zaslání zprávy.

Požadavky

Program implementujte v jazyce C/C++/Python. Pro implementaci je zakázáno využit GitHub Copilot, Chat GPT a jiné AI technologie. Pro implementaci hybridního šifrování můžete využit jakoukoliv knihovnu, mimo knihovny s úplnou implementací této problematiky. V dokumentaci popište vytváření klíčů pro symetrickou a asymetrickou kryptografii, jak jednotlivé klíče používáte a za jakým účelem. Dále popište, jak jste zajistili výměnu klíčů a zajištění integrity. Zaměřte se na jednotlivé kroky předávání zpráv. V dokumentaci taky popište bezpečnost vašeho řešení například:

- Možné prolomení klíče (jak dlouhé klíče používáte? a proč?)
- Narušení integrity, opakované zasílání zpráv, …

Program musí být rychlý, příliš dlouhý výpočet může vést k bodové srážce.

Specifikace knihoven a vlastní práce:
Studenti mohou využít již existující knihovny pro vygenerování kontrolního součtu pomocí algoritmu MD5. Pro symetrické šifrování mohou využít knihovny s implementací algoritmu AES (není potřeba implementovat vlastní generátor náhodných čísel). Klíč mohou získat například z /dev/random, v případně nutnosti /dev/urandom nebo jej vygenerovat pomocí existující knihovny. Pro asymetrické šifrování (RSA) mohou využít již existující řešení pro generování klíčů i pro šifrování/dešifrování.

Pro jednoznačné zarovnání (padding) klíče nebo kontrolního součtu bude potřeba vytvořit vhodný algoritmus bez možnosti využít již existující knihovny. Pro zasílání zpráv studenti mohou využít exitující knihovny, které umožňují zasílat pakety na lokální či jiné síti (V projektu se data budou zasílat pouze na localhost). Kontrolu integrity zprávy a bezpečné odeslání informace, zda paket dorazil v pořádku, tj. Integrita nebyla narušena nebo integrita byla narušena si studenti pohlídají sami bez využití knihovny (mimo knihovny na odesílání paketů).

Makefile
Makefile bude obsahovat minimálně dvě návěští **build** a **run**. Návěští **build** přeloží zdrojový kód ve vámi zvoleném jazyce (pro python zde se program nepřeloží, ale vytvoří se virtuální prostředí s názvem “venv”, následně se prostřední aktivuje pomocí příkazu “source” a instalují se všechny potřebné knihovny ze souboru “requirements.txt”). Návěští **run** spustí program se zadanými parametry.

Příklady spuštění:
Program půjde spustit celkem ve dvou provedení:
1, Klient
2, Server
Program se bude spouštět s několik parametry:
TYPE=c – Program se spustí jako klient
TYPE=s – Program se spustí jako server
PORT=<číslo> - program se spustí na zadaném port**u**
Klient
make run TYPE=c PORT=54321
Server
make run TYPE=s PORT=54321

Zprávy se následně budou vkládat v nekonečné smyčce.

Příklad spuštění a výstupy:

make build
c: make run TYPE=c PORT=54321
s: make run TYPE=s PORT=54321

c: “Successfully connected server” nebo “Error XYZ”
c: RSA_public_key_sender=<veřejný klíč odesílatele>
c: RSA_private_key_sender=<soukromý klíč odesílatele>
c: RSA_public_key_receiver=<veřejný klíč příjemce>

c: **Enter input**: <Zadaný vstup od uživatele>
c: AES_key=<Vygenerovaný/získaný klíč>
c: AES_key_padding=<zarovnaný AES klíč>
c: MD5=<MD5 hash zprávy>
c: MD5_padding=<zarovnaný MD5 hash>
c: RSA_MD5_hash=<Asymetricky zašifrovaný(podepsaný) hash>
c: AES_cipher=<zašifrovaný vstupní text + podepsaný MD5 hash>
c: RSA_AES_key=<Zašifrovaný zarovnaný AES klíč>
c: ciphertext=<Zašifrovaná text + MD5 hash + zašifrovaný klíč>
c: "The message was successfully delivered" nebo "The message was sent again".
c: **Enter input:** <Zadaný vstup od uživatele>
c: …. atd.

s: "Client has joined"
s: RSA_public_key_sender=<veřejný klíč příjemce>
s: RSA_private_key_sender=<soukromý klíč příjemce>
s: RSA_public_key_receiver=<veřejný klíč odesílatele>
s: **ciphertext**=<Zašifrovaný text + MD5 hash + zašifrovaný klíč>
s: RSA_AES_key=<Zašifrovaný a zarovnaný AES klíč>
s: AES_cipher=<zašifrovaný vstupní text + podepsaný MD5 hash>
s: AES_key=<získaný AES klíč>
s: text_hash=<dešifrovaný text a MD5 hash>
s: plaintext=<dešifrovaný text>
s: MD5=<Vygenerovaný MD5 hash>
s: "The integrity of the message has not been compromised." nebo "The integrity of the report has been compromised."
s: **ciphertext**=<Zašifrovaná text + MD5 hash + zašifrovaný klíč>
s: RSA_AES_key=… atd.

Zprávy jsou pouze ilustrativní, můžete je modifikovat ale rozumně!

Hodnocení a testování:

Dokumentace – 2b
Klientská strana – 2.5b
Server strana – 2.5b
Propojení Klient-server – 1b

Zkoumat se bude funkčnost kódu i jeho čitelnost. Kontrola šifrování bude probíhat na úrovni síťových paketů pomocí programu tcp_dump nebo wireshark/t-shark. Dokumentace bude obsahovat

Další poznámky:
- Při implementaci můžete využít libovolnou knihovnu (mimo s hotovou implementací dané problematiky viz Specifikace knihoven a vlastní práce) na serveru merlin.fit.vutbr.cz, na kterém se budou vaše programy testovat.
- Při řešení projektu využijte materiály, které máte poskytnuté v rámci přednášek KRY.
- Další zdroje:

[1] https://digilib.k.utb.cz/bitstream/handle/10563/51862/%C5%BEen%C4%8D%C3%A1k_2022_dp.pdf
[2] https://www.ijcaonline.org/icvci/number9/icvci1377.pdf
[3] https://cs.wikipedia.org/wiki/Hybridn%C3%AD_%C5%A1ifrov%C3%A1n%C3%AD
[4] https://www.researchgate.net/figure/Hybrid-cryptosystem-encryption-decryption-and-digital-signature-Resource-4_fig5_276919918

Odevzdání

Do E-learningu (Moodlu) odevzdávejte archiv .zip pojmenovaným vašim osobním číslem například 1233456.zip se zdrojovými kódy vaší aplikace a dokumentací ve formátu .pdf. Aplikace musí obsahovat makefile, který po zavolání příkazu "make build" vytvoří spustitelný soubor "kry" (v případě jazyku python kry.py).

- Projekt odevzdávejte ve formátu ZIP (TAR, 7Zip, RAR a jiné == 0 bodů)
- Využití technologie Chat GPT, Github Copilot a jiné AI technologie == 0 bodů
- Archiv pojmenujte Vašim osobním číslem: 123456.zip
- Archiv v sobě nebude obsahovat žádné složky (mimo složky “cert”), složky pojmenované Vašim loginem, src, a podobné == 0 bodů
- Makefile vytvoří program s názvem ./kry (v případě jazyku python kry.py)
- Dokumentace se zaměří na problematiku šifrování/dešifrování/zarovnání/integrita, předávání zpráv a bezpečností analýza. Dokumentace se bude odevzdávat ve formátu .pdf.
- Během běhu nevypisujte na stdout zbytečnosti, případné chyby vypisujte na stderr.
- Všechny zadané vstupy při testování budou korektní, není potřeba kontrolovat zadané argumenty nebo formát vstupů.

Konzultace k projektu poskytuje Ing. Daniel Snášel: isnasel@fit.vutbr.cz
Datum odevzdání: 23.4.2023 23:59:59