Úvod do počítačovej bezpečnosti

Zadanie 3 - Implementácia aplikácie na šifrovanie súborov s využitím RSA a kontroly integrity

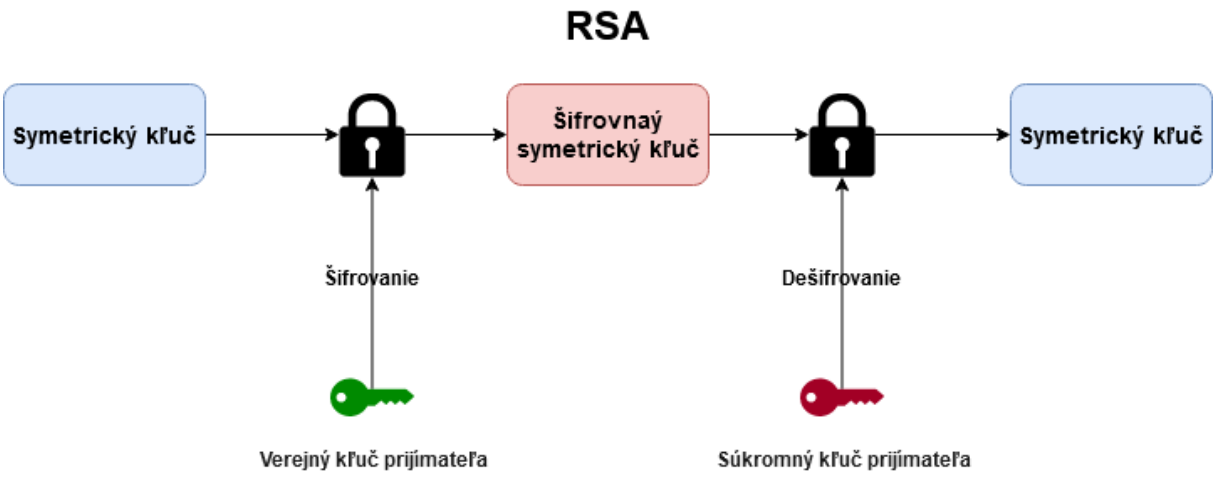
Cieľom zadanie bolo rozšíriť aplikáciu z 2. zadanie o kontrolu integrity, pričom kľúč (zašifrovaný asymetricky RSA) a IV majú byť posielané v rámci hlavičky šifrovaného súboru.

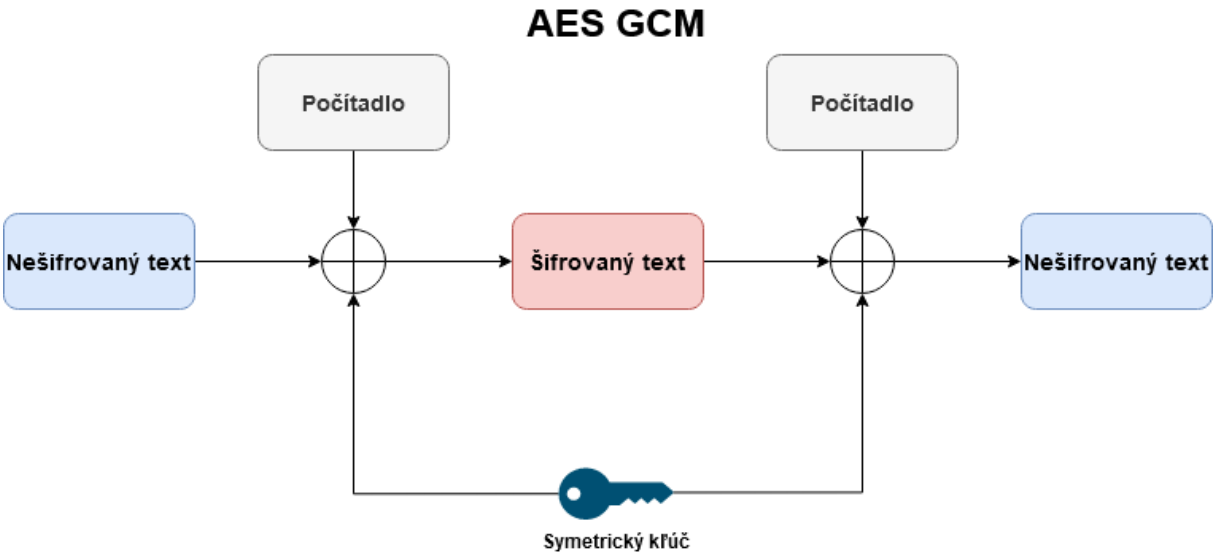
A. Špecifikácia

- Programovací jazyk: Java
- Knižnice: `java.security` (security framework)
  `javax.crypto` (kryptografické operácie)
- Zdroje: https://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html

B. Použitá šifra

Aplikácia využíva na šifrovanie šifrovací algoritmus AES v GCM móde s kľúčom veľkosti 256 bitov. AES-GCM je prevádzkový režim blokovej šifry, ktorý poskytuje vysokú rýchlosť autentifikovaného šifrovania a kontrolu integrity údajov. V režime GCM sa šifrovanie blokov transformuje na šifrovanie toku, a preto nie je potrebný žiaden PADDING. Dodatočné overovacie údaje (Authenticated Encryption with Associated Data AEAD a Authenticated Associated Data AAD) nie sú šifrované a používajú sa pri výpočte overovacej značky. Operácia autentifikovaného šifrovania vyžaduje Initialization Vector (IV), Additional Authenticated Data (AAD) a tajný kľúč pričom produkuje zašifrovaný text a autentifikačný štítok. Tajný kľúč použitý na šifrovanie správy je šifrovaný RSA šifrou a priložený spolu s IV do hlavičky zašifrovaného súboru. Pri dešifrovaní súboru dochádza ku kontrole autenticity a integrity dát, v prípade, že kontrola zlyhá aplikácia hodí chybovú hlášku v dôsledku čoho bude vytvorený prázdny dešifrovaný súbor.





C.Spustenie aplikácie

Aby bolo možné aplikáciu spustiť je potrebné mať na zariadení nainštalovanú JAVU. Aplikácia nemá vlastné grafické používateľské rozhranie (GUI), preto je pre jej spustenie potrebné otvoriť konzolu, prejsť do adresára kde sa nachádza zadanie3.jar súbor a príkazom **`java -jar zadanie3.jar`** (Windows, Mac OS, Linux) aplikáciu spustiť.

D.Používateľská príručka

1. Návod na použitie aplikácie

```
D:\Zadanie3>java -jar zadanie3.jar
Aplikácia bola úspešne spustená...

Zadajte názov resp. absolútnu cestu k súboru (Nešifrovaný súbor):
```

Po úspešnom spustení aplikácie je užívateľ vyzvaný zadať názov súboru, ktorý sa má zašifrovať. V prípade, že sa súbor nachádza v rovnakom adresári stačí zadať názov inak je potrebné uviesť absolútnu cestu k súboru aby ho aplikácia vedela nájsť.

```
Zadajte názov resp. absolútnu cestu k súboru (Nešifrovaný súbor):
Súbor sa nepodarilo nájsť, skúste zadať názov súboru ešte raz:
```

V prípade, že súbor nebude možné nájsť, aplikácia Vám umožní názov zadať opätovne.

```
Zadajte názov resp. absolútnu cestu k súboru (Nešifrovaný súbor): test.txt

Prebieha šifrovanie...
Šifrovanie bolo dokončené (2 ms)...
Šifrovaný súbor 'D:\Zadanie3\test.encrypted' bol vytvorený.

Pre dešifrovanie stlačte ENTER...
```

Ak aplikácia súbor nájde a bude ho vedieť otvoriť automaticky začne so šifrovaním. Po skončení šifrovania bude vytvorený súbor: **<názov>.encrypted**, ktorý reprezentuje šifrovaný súbor. Pre pokračovanie na proces dešifrovanie je potrebné stlačiť ENTER.

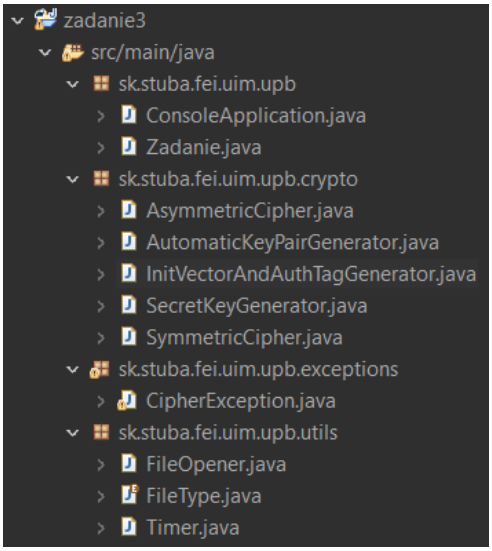
```
Pre dešifrovanie stlačte ENTER...

Zadajte názov resp. absolútnu cestu k súboru (Šifrovaný súbor): test.encrypted

Prebieha dešifrovanie...
Dešifrovanie bolo dokončené (244600 ns)...
Dešifrovaný súbor 'D:\Zadanie3\test.decrypted' bol vytvorený.
```

Pre dešifrovanie je potrebné zadať názov respektíve cestu k zašifrovanému súboru (**je nutné aby mal zašifrovaný súbor príponu .encrypted** inak aplikácia bude od Vás súbor vyžadovať opätovne). Po skončení dešifrovania bude vytvorený súbor **<názov>.decrypted**. Tento súbor by mal byť zhodný so vstupným súborom, ktorý sme šifrovali. Pre skontrolovanie zhody stačí len manuálne prepísať príponu súboru na takú akú mal pôvodný súbor a prezrieť si obsah.

2. Návod na importovanie projektu do IDE

Aplikácia bola vyvíjaná v Eclipse IDE pričom bola vytvorená ako Gradle projekt, pre prípad, že by ste chceli otvárať kód aplikácie v IDE odporúčam aplikáciu importovať ako „existing gradle project“ aby náhodou nevznikali nejaké komplikácie. V prípade, že by ste si chceli nanovo vygenerovať .jar stačí použiť gradle task „`jar`“, ktorý celý kód skompiluje vybuilduje a zabalí do `zadanie3.jar`, takto vygenerovaný súbor je potom možné nájsť v `zadanie3\build\libs` adresári a spustiť ho.

3. Štruktúra projektu



4. Stručný popis hlavných tried:

Trieda	Funkcionalita
ConsoleApplication	Hlavná trieda aplikácie
FileOpener	Vyhľadáva a otvára súbory
SymmetricCipher	Šifrovanie a dešifrovanie súborov
AsymmetricCipher	Šifrovanie a dešifrovanie kľúča
SecretKeyGenerator	Generátor náhodného šifrovacieho kľúča
AutomaticKeyPairGenerator	Generátor verejného s súkromného kľúča
InitVectorAndAuthTagGenerator	Generátor inicializačného vektora a tagu na kontrolu integrity
Timer	Časovač šifrovania/dešifrovanie

Symetrická šifra (SymmetricCipher)

- Na šifrovanie bola použitá trieda `javax.crypto.Cipher`, ktorá poskytuje funkcionalitu na šifrovanie a dešifrovanie konkrétnych kryptografických šifier
- Šifrovací algoritmus: AES/GCM/NoPadding

Asymetrická šifra (AsymmetricCipher)

- Na šifrovanie bola použitá trieda `javax.crypto.Cipher`, ktorá poskytuje funkcionalitu na šifrovanie a dešifrovanie konkrétnych kryptografických šifier
- Šifrovací algoritmus: RSA/ECB/OAEPWithSHA-256AndMGF1Padding
- Dešifrovací algoritmus: RSA/ECB/OAEPPadding

Tajný kľuč (SecretKeyGenerator)

- Na generovanie šifrovacieho kľúča bola využitá trieda `javax.crypto.KeyGenerator`, ktorá poskytuje funkcionalitu na generovanie tajných symetrických kľúčov.
- Na to aby bol kľuč silný, bezpečný a naozaj náhodný bola využitá trieda `java.security.SecureRandom`, ktorá poskytuje kryptograficky silný generátor náhodných čísel.
- Algoritmus generovaného kľúča je AES s veľkosťou 256 bitov

Kľúčový pár (AutomaticKeyPairGenerator)

- Na generovanie kľúčového páru bola využitá trieda `java.security.KeyPairGenerator`, ktorá poskytuje funkcionalitu na generovanie verejného a súkromného kľúča.
- Na to aby bol kľuč silný, bezpečný a naozaj náhodný bola využitá trieda `java.security.SecureRandom`, ktorá poskytuje kryptograficky silný generátor náhodných čísel.
- Algoritmus generovaného kľúčového páru je RSA s veľkosťou 2048 bitov

Inicializačný vektor a tag na kontrolu integrity (InitVectorAndAuthTagGenerator)

- Na generovanie inicializačného vektora a tagu na kontrolu integrity dát bola využitá trieda `javax.crypto.spec.GCMParameterSpec`.
- Veľkosť inicializačného vektora je 12 bajtov a veľkosť tagu je 16 bajtov.
- Do tagu bol pridaný aj zašifrovaný symetricky kľuč, IV nie je nutné manuálne pridávať, pretože sa pridáva automaticky cez **GCMParameterSpec.class**

E. Bezpečnostné odporúčania

- Nezverejňovať kľúč tretím stranám a neoprávneným osobám
- Nezanechávať kľúč na rôznych úložiskách (HDD, USB,...) bez zabezpečenia
- Pri šifrovaní je doležíte klásť dôraz na bezpečnostné odporúčania