

Комп'ютерний приктикум номер 2
**«Побудова протоколу цифрового підпису на основі
схеми Ель-Гамалю та її варіантів»**

Варіант 10

Виконали:

Студенти групи ФІ-43

Мариненко Артем

Кривцун Владислав

Перевірив:

Фесенко А.В

Київ 2018

Мета

Реалізація асиметричної криптосистеми на основі схеми цифрового підпису Ель-Гамала. Знайомство з організацією криптографічних протоколів та складовими елементами цих протоколів такими як , функції хешування , шифруючі перетворення , тощо.

Засоби

Для написання комп'ютерного практикуму було використано мову Java ,
Та її бібліотеку великих чисел BigInteger.

Хід роботи

Для успішного написання цієї роботи необхідно послідовно реалізовувати потрібний нам функціонал. В першу чергу написати шифруючі перетворення.

Шифруюче перетворення

Використовується блочний шифр $E_K(M)$, який перетворює 64 бітні блоки на 64 бітному ключі. Шифр є чотирьохраундовою фейстель-подібною схемою(MISTY).

1)Вхідний блок розбивається на дві половини по 32 біта L_0 та R_0

$$M = L_0 || R_0$$

У комп'ютерному практикумі для коректної роботи алгоритму було необхідно зареверсити байти в L_0 та R_0

2)Для $i=1,2,3,4$ обчислюється

$$L_i = (S(K_i \oplus R_{i-1}) \ll 13) \oplus L_{i-1}$$

$$R_i = L_{i-1}$$

3)Шифртекстом є конкатенація $C = R_4 || L_4$

$S(X)$ –Нелінійне перетворення , таблиця підстановки.

Формування раундових ключів відбувається наступним чином K
розбивається на дві половини по 32 біта $K = K_L || K_R$, в лабораторній роботі

ми ключ K був представлений у вигляді масиву байт , тобто нам треба було розділити ключ на два масиви по 4 байти кожен.

Далі самі раундові ключі формуються так $K_1 = K_L, K_2 = K_R, K_3 = \overline{K_R},$

$$K_4 = \overline{K_L}$$

Весь вище описаний процес ми можемо бачити у коді, у функції *encryption()*

Далі ми підходимо до реалізації геш функції

Геш-функція

На вхід функції подається повідомлення M , від якого нам необхідно обчислити геш, ми перевіряємо чи кратна його довжина 64 , якщо ні то додаємо одиничний біт та кількість нулів якої не достає для виконання умови.

Далі розбиваємо повідомлення на блоки по 64 біти або 8 байт.

У коді можна знайти як функцію *addPadding()*

1)

$$M = M_1 || M_2 || \dots || M_t$$

2)

Після розбиву проводяться ітерації $i=1,2,3,\dots, t$

$$H_i = G(M_i, H_{i-1})$$

H_0 -64 нульових біта , а функція $G(M, H) = E_{M \oplus H}(H) \oplus H$

Можна знайти у лабораторній як *gFunction()*

3)Гешем повідомлення є значення H_t

У коді функцію хешування можна знайти як *hashFunction()*

Тепер можна взятися за реалізацію цифрового підпису

Цифровий підпис

У лабораторній роботі необхідно реалізувати цифровий підпис Ель-Гамала,

Стійкість якого ґрунтується на проблемі обчислення дискретного логарифму

.

Загальносистемні параметри :

- $q = \text{AF5228967057FE1CB84B92511BE89A47}$

- $p = \text{57A9144B382BFF0E5C25C9288DF44D23}$

- $a = \text{9E93A4096E5416CED0242228014B67B5}$

x - довільне число менше p

$$y = a^x \bmod p$$

Підписом під повідомленням с гешем h є пара (k, S) , що знаходиться за наступним алгоритмом :

1)Геш h перетворюється на велике число H

2) Обирається випадкове число U

3)Обчислюється число Z : $Z = a^U \bmod p$

4)По значенням H, U, Z та ключів k, g

5)Обчислюється число S : $S = a^g \bmod p$

H представимо у такому вигляді де молодші індекси це молодші байти гешу

$h / 1 h / 2 h / 3 h / 4 h / 5 h / 6 h / 7 h / 00 / FF / FF / FF / FF / FF / FF / 00$

В нашому варіанті ключи обчислюються за наступними формулами

$$k = \frac{(UH - xZ)}{H} \bmod q, g = \frac{xZ}{H} \bmod q$$

Та з перевірочним виразом $S^H = y^{(a^{kS \bmod p})} \bmod p$

Опис роботи програми(короткий)

Спочатку зчитуємо файл у масив байтів за допомогою функції `readAllBytes()`

Далі додаємо паддінг

Потім предаємо цей масив у `hashFunction()`

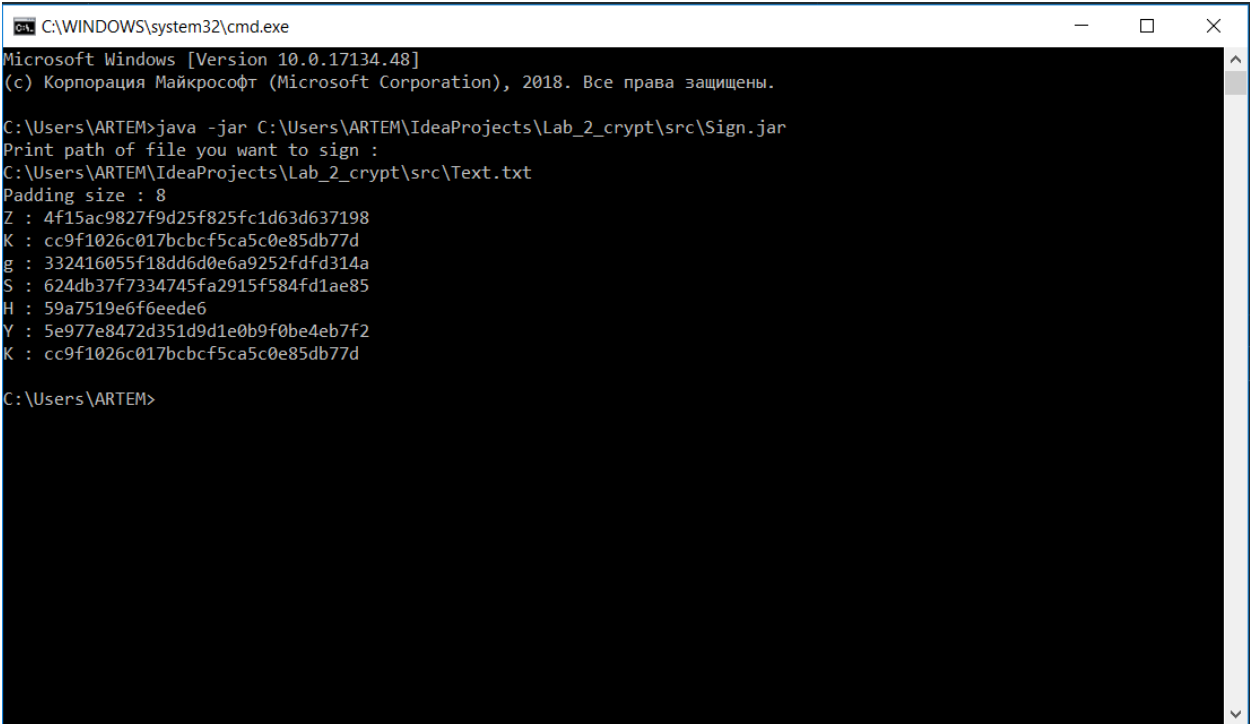
Далі передаємо результат хешування у вигляді масиву байтів у функцію *signature()* де відбувається обчислювання підпису.

Як використовувати програму

Для постановки підпису із командної стрічки потрібно набрати наступний код:

```
java -jar шлях до jar файлу з назвою Sign.jar
```

далі виконувати інструкції які побачите у командній стрічці , потрібно ввести шлях до файлу , який хочете підписати.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.17134.48]
(c) Корпорация Майкрософт (Microsoft Corporation), 2018. Все права защищены.

C:\Users\ARTEM>java -jar C:\Users\ARTEM\IdeaProjects\Lab_2_crypt\src\Sign.jar
Print path of file you want to sign :
C:\Users\ARTEM\IdeaProjects\Lab_2_crypt\src\Text.txt
Padding size : 8
Z : 4f15ac9827f9d25f825fc1d63d637198
K : cc9f1026c017bcbcf5ca5c0e85db77d
g : 332416055f18dd6d0e6a9252fd314a
S : 624db37f7334745fa2915f584fd1ae85
H : 59a7519e6f6eede6
Y : 5e977e8472d351d9d1e0b9f0be4eb7f2
K : cc9f1026c017bcbcf5ca5c0e85db77d

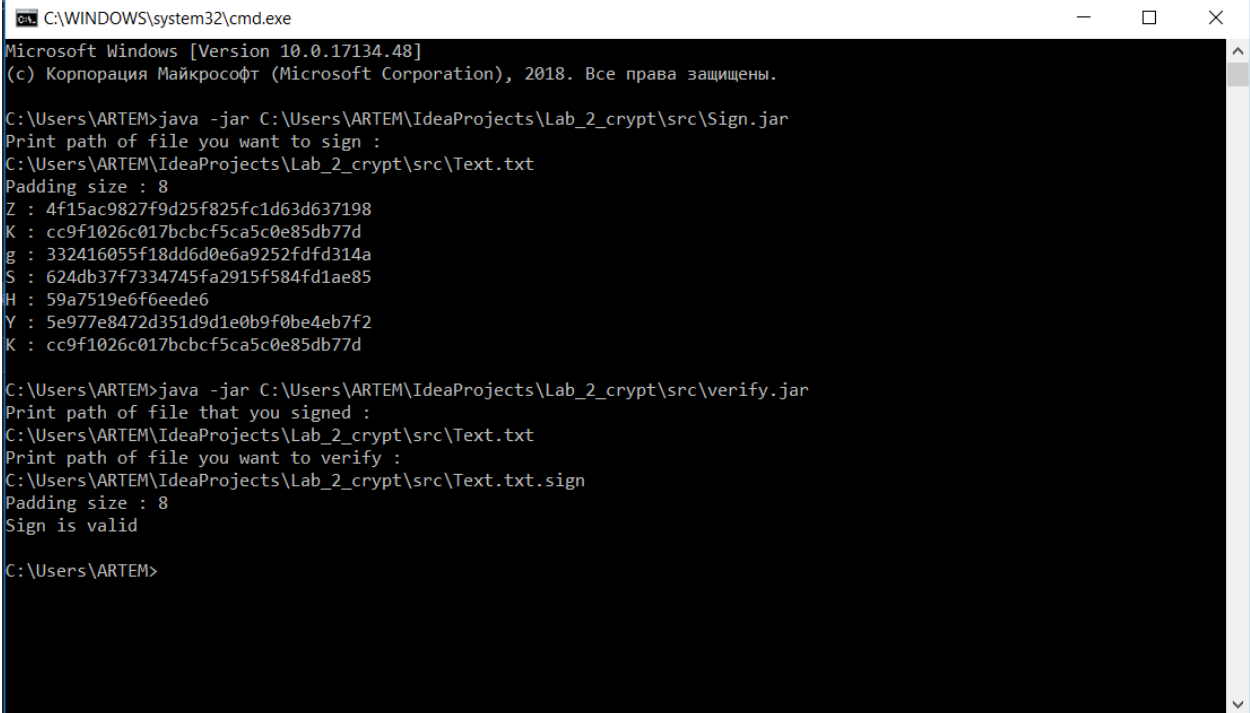
C:\Users\ARTEM>
```

Для перевірки підпису треба набрати наступну команду у командній стрічці

java -jar шлях до jar файлу з назвою verify.jar

ввести шлях до файлу який підписувався та до файлу звіту сформованого програмою підпису, якщо підпис правильний ви побачите повідомлення

«Sign is valid»



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.17134.48]
(c) Корпорация Майкрософт (Microsoft Corporation), 2018. Все права защищены.

C:\Users\ARTEM>java -jar C:\Users\ARTEM\IdeaProjects\Lab_2_crypt\src\Sign.jar
Print path of file you want to sign :
C:\Users\ARTEM\IdeaProjects\Lab_2_crypt\src\Text.txt
Padding size : 8
Z : 4f15ac9827f9d25f825fc1d63d637198
K : cc9f1026c017bcbcf5ca5c0e85db77d
g : 332416055f18dd6d0e6a9252fd314a
S : 624db37f7334745fa2915f584fd1ae85
H : 59a7519e6f6eede6
Y : 5e977e8472d351d9d1e0b9f0be4eb7f2
K : cc9f1026c017bcbcf5ca5c0e85db77d

C:\Users\ARTEM>java -jar C:\Users\ARTEM\IdeaProjects\Lab_2_crypt\src\verify.jar
Print path of file that you signed :
C:\Users\ARTEM\IdeaProjects\Lab_2_crypt\src\Text.txt
Print path of file you want to verify :
C:\Users\ARTEM\IdeaProjects\Lab_2_crypt\src\Text.txt.sign
Padding size : 8
Sign is valid

C:\Users\ARTEM>
```

Висновки

Під час виконання лабораторної роботи було реалізовано схему цифрового підпису типу Ель-Гамалія . Генератором псевдо випадкових чисел було обрано генератор java бібліотеки Random, так як у тестах минулого семестру він показав гарні результати. Вдалося уникнути труднощів при реалізації алгоритму цифрового підпису.