**УКРАЇНИ МІНІСТЕРСТВО ОСВІТИ І НАУКИ**

**СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ**

**ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ**

**КАФЕДРА КОМП’ЮТЕРНИХ НАУК**

**ПРАКТИЧНА РОБОТА № 10**

з дисципліни "Захист інформації"

Тема: "**Електронний цифровий підпис. ЕЦП на основі RSA**"

Варіант № 14

**Перевірив Коробченко О.В.**

**Студентка групи Індн – 31с Пархомчук А. К.**

Суми – 2017

**Мета**

Виробити у студента вміння та навички генерації та перевірки ЕЦП на основі алгоритму RSA.

**Завдання**

1. Ознайомитися з теоретичними відомостями про ЕП

2. Виконати програмну реалізацію генерації ЕЦП і перевірку ЕЦП на будь-якій алгоритмічній мові.

3. Протестувати роботу програми.

4. Оформити звіт.

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **A** | **B** |
| 14 | 16500 | 24000 |

**Теоретичний матеріал**

**Схема роботи алгоритму MD5**

На вхід алгоритму надходить вхідний потік даних, хеш якого необхідно знайти, а на виході алгоритму отримуємо 128-бітове хеш значення. Довжина повідомлення може бути будь-якою (у тому числі нульовою). Запишемо довжину повідомлення в L. Це число ціле і не негативне. Кратність яким-небудь числам не обов'язкова. Після надходження даних йде процес підготовки потоку до обчислень.

Нижче наведено 5 кроків алгоритму:

**Крок 1. Вирівнювання потоку**

Вхідні дані вирівнюються так, щоб їх розмір був порівнянний з 448 по модулю 512 (L’ = 512 × N + 448). Спочатку дописують одиничний біт в кінець потоку, потім необхідне число нульових біт (вирівнювання відбувається, навіть якщо довжина вже конгруентна - порівнянна з 448).

**Крок 2. Додавання довжини повідомлення**

В останні 64 біта дописують 64-бітове представлення довжини даних до вирівнювання. Якщо довжина перевершує 264 − 1, то дописують тільки молодші біти. Після цього довжина потоку стане кратна ступеням двійки — 16, 32. Обчислення будуть ґрунтуватися на уявленні цього потоку даних у вигляді масиву слів по 512 біт (розбитими на 16 32-бітових підблоків).

**Крок 3. Ініціалізація буфера**

Для обчислень ініціалізуються 4 змінних розміром по 32 біта і задаються початкові значення шістнадцятковими числами:

*A* =0x01234567

*B* = 0x89abcdef

*C* = 0xfedcba98

*D* =0x76543210

У цих змінних будуть зберігатися результати проміжних обчислень. Початковий стан ABCD називається вектором ініціалізації.

**Крок 4. Обчислення в циклі**

Чотири змінних копіюються в інші змінні: A в a, B в b, C в c и D в d.)

Головний цикл складається з чотирьох дуже схожих. На кожному етапі 16 раз використовуються різні операції. Кожна операція являє собою лінійну функцію над трьома з *a*, *b*, *c* і *d*. Потім вона додає цей результат до четвертої змінної, підблоку тексту і константі. Далі результат циклічно зсувається вправо на змінне число бітів і додає результат до однієї з змінних *a*, *b*, *c* і *d*. Врешті результат замінює одну із змінних *a*, *b*, *c* і *d*. Існують чотири нелінійних функції, використовувані по одній в кожній операції (для кожного етапу - інша функція).

1. Fun F(X,Y,Z) = (X\wedge{Y}) \vee (\neg{X} \wedge{Z})
2. Fun G(X,Y,Z) = (X\wedge{Z}) \vee (\neg{Z} \wedge{Y})
3. Fun H(X,Y,Z) = X \oplus Y \oplus Z
4. Fun I(X,Y,Z) = Y \oplus (\neg{Z} \vee X)

Якщо  позначає *j*-ий підблок повідомлення (від 0 до 15), а <<<s означає циклічний зсув вліво на s бітів, то використовуються наступні чотири операції :

FF(a,b,c,d,Mj,s,ti) означає a = *b* + ((*a* + F(b,c,d) + Mj + ti,) <<<s) GG(a,b,c,d,Mj,s,ti) означає a = b + ((*a* + *G*(*b,c,d*) + + Mj + ti) <<<s) *HH*(a,b,c,d,Mj,s,ti,) означає a = *b* + ((*a* + *H*(*b*,*c*,*d*) + + Mj + ti) <<<s) II(a,b,c,d,Mj,s,ti) означає a = b + ((*a* + *I*(*b*,*c*,*d*) + + Mj + ti) <<<s)

Чотири етапи (64 дії виглядають наступним чином):

**Етап 1**:

1. FF(a, b, c, d, M0, 7, 0xd76aa478)
2. FF(d, a, b, c, M1, 12, 0xe8c7b756)
3. FF(c, d, a, b, M2, 17, 0x242070db)
4. FF(b, c, d, a, M3, 22, 0xc1bdceee)
5. FF(a, b, c, d, M4, 7, 0xf57c0faf)
6. FF(d, a, b, c, M5, 12, 0x4787c62a)
7. FF(c, d, a, b, M6, 17, 0xa8304613)
8. FF(b, c, d, a, M7, 22, 0xfd469501)
9. FF(a, b, c, d, M8, 7, 0x698098d8)
10. FF(d, a, b, c, M9, 12, 0x8b44f7af)
11. FF(c, d, a, b, M10, 17, 0xffff5bb1)
12. FF(b, c, d, a, M11, 22, 0x895cd7be)
13. FF(a, b, c, d, M12, 7, 0x6b901122)
14. FF(d, a, b, c, M13, 12, 0xfd987193)
15. FF(c, d, a, b, M14, 17, 0xa679438e)
16. FF(b, c, d, a, M15, 22, 0x49b40821)

**Етап 2**

1. GG(a, b, c, d, M1, 5, 0xf61e2562)
2. GG(d, a, b, c, M6, 9, 0xc040b340)
3. GG(c, d, a, b, M11, 14, 0x265e5a51)
4. GG(b, c, d, a, M0, 20, 0xe9b6c7aa)
5. GG(a, b, c, d, M5, 5, 0xd62f105d)
6. GG(d, a, b, c, M10, 9, 0x02441453)
7. GG(c, d, a, b, M15, 14, 0xd8ale681)
8. GG(b, c, d, a, M4, 20, 0xe7d3fbc8)
9. GG(a, b, c, d, M9, 5, 0x2,lelcde6)
10. GG(d, a, b, c, M14, 9, 0xc33707d6)
11. GG(c, d, a, b, M3, 14, 0xf4d50d87)
12. GG(b, c, d, a, M8, 20, 0x455al4ed)
13. GG(a, b, c, d, M13, 5, 0xa9e3e905)
14. GG(d, a, b, c, *M2,* 9, 0xfcefa3f8)
15. GG(c, d, a, b, M7, 14, 0x676f02d9)
16. GG(b, c, d, a, M12, 20, 0x8d2a4c8a)

**Етап 3**:

1. HH(a, b, c, d, M5*,* 4, 0xfffa3942)
2. HH(d, a, b, c*,* M8, 11, 0x8771f681)
3. HH(c, d, a, b, M11, 16, 0x6d9d6122)
4. HH(b, c, d, a, M14, 23, 0xfde5380c)
5. HH(a, b, c, d, M1, 4, 0xa4beea44)
6. HH(d, a, b, c, M4, 11, 0x4bdecfa9)
7. HH(c, d, a, b, M7, 16, 0xf6bb4b60)
8. HH(b, c, d, a, M10, 23, 0xbebfbc70)
9. HH(a, b, c, d, M13, 4, 0x289b7ec6)
10. HH(d, a, b, c, M0, 11, 0xeaa127fa)
11. HH(c, d, a, b, M3, 16, 0xd4ef3085)
12. HH(b, c, d, a, M6, 23, 0x04881d05)
13. HH(a, b, c, d, M9, 4, 0xd9d4d039)
14. HH(d, a, b, c, M12, 11, 0xe6db99e5)
15. HH(c, d, a, b, M15, 16, 0x1fa27cf8)
16. HH(b, c, d, a, M2, 23, 0xc4ac5665)

**Етап 4**:

1. II(a, b, c, d, M0, 6, 0xf4292244)
2. II(d, a, b, c, M7, 10, 0x432aff97)
3. II(c, d, a, b, M14, 15, 0xab9423a7)
4. II(b, c, d, a, M5, 21, 0xfc93a039)
5. II(a, b, c, d, M12, 6, 0x655b59c3)
6. II(d, a, b, c, M3, 10, 0x8f0ccc92)
7. II(c, d, a, b, M10, 15, 0xffeff47d)
8. II(b, c, d, a, M1, 21, 0x85845ddl)
9. II(a, b, c, d, M8, 6, 0x6fa87e4f)
10. II(d, a, b, c, M15, 10, 0xfe2ce6e0)
11. II(c, d, a, b, M6, 15, 0xa3014314)
12. II(b, c, d, a, M13, 21, 0x4e081lal)
13. II(a, b, c, d, M4, 6, 0xf7537e82)
14. II(d, a, b, c, M11, 10, 0xbd3af235)
15. II(c, d, a, b, M2, 15, 0x2ad7d2bb)
16. II(b, c, d, a, M9, 21, 0xeb86d391)

Після всього цього a, b, c і d додаються до A, B, C і D, відповідно, і алгоритм продовжується для наступного 512-бітного блоку даних (до повного їх вичерпання).

**Крок 5. Результат обчислень**

Результат обчислень знаходиться в буфері ABCD, це і є хеш. Якщо вивести слова в зворотному порядку DCBA, то ми отримаємо наш MD5 хеш.

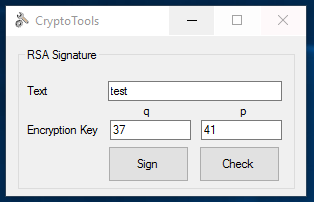
**Вихідні дані**

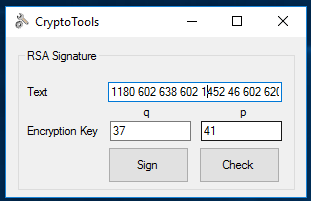
Q=37

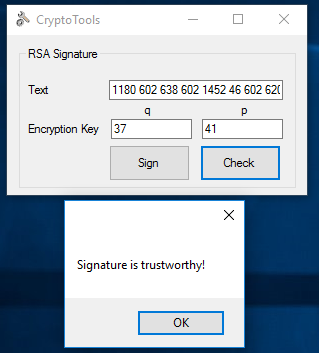
P=41

Text=test

**Скриншоти роботи програми**







**Вихідний текст програми**

Додається до звіту

**Результати роботи програми**

Q=37

P=41

Result = 1180 602 638 602 1452 46 602 620 602 602 953 602 519 389 602 602

**Висновки**

При виконанні практичної роботи вироблені навички генерації та перевірки ЕЦП на основі алгоритму RSA.