НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичних методів захисту інформації

Звіт до
комп'ютерного практикуму №3

Оформлення звіту:
Дигас Богдан, ФІ-52мн
Юрчук Олексій, ФІ-52мн

6 жовтня 2025 р.
м. Київ

Комп'ютерний практикум № 1

1.1 Вступні відомості

Мета роботи: Ознайомлення з підходами побудови атак на асиметричні криптосистеми на прикладі атак на криптосистему RSA, а саме атаки на основі китайської теореми про лишки, що є успішною при використанні однакового малого значення відкритої експоненти для багатьох користувачів та атаки "зустріч посередині", яка можлива у випадку, якщо шифротекст є невеликим числом, що є добутком двох чисел.

Постановка задачі:

1. Створіть репозиторій у системі контролю версій Git/GitHub;
2. Реалізувати атаку з малою експонентою на основі китайської теореми про лишки.
3. Реалізувати атаку "зустріч посередині" та порівняти її швидкодію з повним перебором можливих відкритих текстів.
4. Оформити звіт до комп'ютерного практикуму.

Результати виконання роботи. Варіант 15

1.2 Small exponent attack

1.2.1 Принцип алгоритму:

Користувач надсилає однакове повідомлення M кільком k користувачам $A_1, A_2, \dots, A_k$, $k \in \mathbb{N}$, причому відкритий ключ користувача $A_i, i = \overline{1,k}$ – це пара чисел (n_i, e), де e – деяке мале число. Тобто юзери $A_1, A_2, \dots, A_k$ мають однакову експоненту e, яка використовується для шифрування. Тоді відправник зашифровує своє повідомлення M, використовуючи відповідні пари (n_i, e) для кожного з отримувачі $A_i, i = \overline{1,k}$.

Зловмисник підключається до каналу передачі даних і перехоплює шифротексти $C_1, C_2, \dots, C_k$, де $C_i = M^e \mod n_i, i = \overline{1,k}$ Вважаємо, що значення n_i для $i = \overline{1,k}$ попарно взаємно прості. (Інакше можна було б обчислити gcd і факторизувати, як мінінум, два модуля $n_i, n_j, i \neq j$, обчислюючи $gcd(n_i, n_j)$). А також, що $M < \min\{n_i | i = \overline{1,k}\}$. Тобто маємо такий випадок:

$$\begin{cases} C_1 = M^e \mod n_1; \\ C_2 = M^e \mod n_2; \\ \dots \\ C_k = M^e \mod n_k. \end{cases}$$

Якщо $k \geq e$, то зловмисник, перехопивши значення $C_1, \ldots, C_k$, може дешифрувати повідомлення M наступним чином:

1. Обчислює значення $C = M^e \mod (n_1 \cdot n_2 \cdot \ldots \cdot n_k)$, використовуючи китайську теорему про лишки;
2. Оскільки $M < \min\left\{n_i | i = \overline{1,k}\right\}$, то $M^e < n_1 \cdot n_2 \cdot \ldots \cdot n_k$. Тоді $C = M^e$.
3. Обчислюємо корень m-того степеня і знаходимо оригінальне повідомлення: $M = \sqrt[e]{C}$

1.2.2 Results of SE

Згідно варіанту було взято спершу тестовий dummy-варіант, а потім, власне, real завдання. Також для порівняння часу виконання, було реалізовано додаткову функцію яка обраховує корені n-того степеня методом Ньютона (метод дотичних).

Отримані наступні результати:

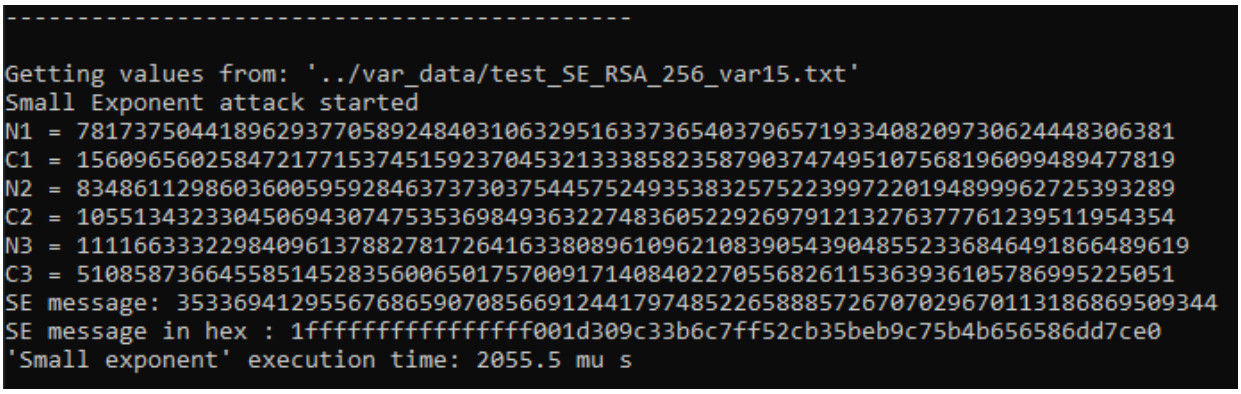


Рис. 1.1: На тестових даних

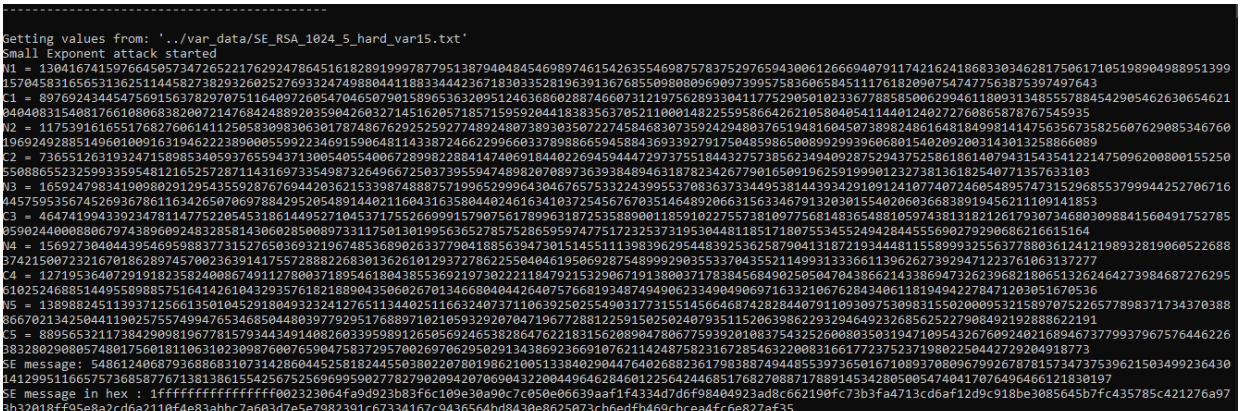


Рис. 1.2: На стандартному варіанті, з custom $\sqrt[e]{C}$

Рис. 1.3: На стандартному варіанті, з inbuild $\sqrt[e]{C}$

Для даного виду атаки, різниця часу у зв'язку з застовуванням іншої функції обчислення $\sqrt[e]{C}$ не є суттєвою, вона скаладає $\sim 50000\,\mu s$, що дорівнює усього $0.05\,s$. Як можна бачити, стандартна реалізація є більш вдалою.

1.3 Man(Meet) in the middle attack

1.3.1 Принцип алгоритму:

Зловмисник перехопив деякий шифротекст $C = M^e \mod n$, причому відомо, що $M < 2^l$, $l \ll \log_2 n$. З доволі великою ймовірністю повідомлення M є складеним числом, тобто його можна представити як добуток чисел $M_1 \cdot M_2$. Нехай при цьому $M_1 \leq 2^{l/2}$ та $M_2 \leq 2^{l/2}$. Тобто можемо розписати:

$$C = (M_1 \cdot M_2)^e \mod n = M_1^e \cdot M_2^e \mod n.$$

Зловмисник дешифровує повідомлення M, виконуючи наступні кроки.

1. Криптоаналітик формує множину пар X виду:

$$X = \left\{(1,1), (2, 2^e \mod n), \ldots, \left(2^{l/2}, \left(2^{l/2}\right)^e \mod n\right)\right\}$$

   Кожна пара має вигляд: $(T, T^e \mod n)$, де $T = \overline{1, 2^{l/2}}$

2. Далі послідовно обчислює такі значення:

$$C_S = C \cdot S^{-e} \mod n, \quad S = \overline{1, 2^{l/2}},$$

   де $S^{-e} \mod n$ – попередньо обраховані значення з множини X.

   (а) Для кожного значення C_S, одразу після його обчислення, зловмисник шукає в множині X таку пару, щоб $S = (T^e \mod n)$ для деякого значення $T = \overline{1, 2^{l/2}}$.

   (б) Якщо таке T не знайдено, повертаємось на крок 2 та обчислюємо наступне значення C_{S+1}. Якщо при цьому $S = 2^{l/2}$, то алгоритм дешифрування зупиняє роботу і виводить *"Відкритий текст не визначено"*.

3. Для знайденого значення $T^e \mod n$ виконується рівність:

$$T^e = C \cdot S^{-e} \mod n.$$

Тоді маємо:

$$C = (T \cdot S)^e \mod n,$$

тобто шифротекст C було отримано внаслідок шифрування відкритого тексту $T \cdot S$

1.3.2 Results of MitM

Також згідно варіанту було взято тестовий dummy-варіант для MitM, а потім, власне, real завдання. Отримані наступні результати:

```
-------------------------------------------

Getting values from: '../var_data/test_MitM_RSA_256_var15.txt'
> MitM (optimized) attack started for:
l = 56
N: 100039008232565778030511827934443693881873768818632993921733760872111379080792863325428540036282008104493965528664727
997929298663632421223589042362967890 77
C: 37870606681322641487557904725248729528324340829502713846646659955496503888917562734432487838642304582405282356780160 1
109059756289250418463739605461441913 6
Start: bn_t = 0, bn_s = 0 : 71
Generated: bn_t = 0, bn_s = 0 : 190958104
MitM message: 425079
MitM message in hex : 67c77
Optimized 'Meet in the middle' execution time: 2.85559e+08 mu s
```

```
-------------------------------------------

Getting values from: '../var_data/MitM_RSA_2048_20_regular_hard_var15.txt'
> MitM (optimized) attack started for:
l = 56
N: 259184146537004307610829237988106804772837188562507139305915304689282063752456996295814599082339878915899244151969020
997425935706030233676545600833751815080442497423996571705433224827129481638564294857125889017489932948312704767428346365
643592537205591638640693079296429124619892458053799110404126860354066048967963210822899570932576193549612174086962466286
300921682880177151937211376971952280544893345324682201719617813244216767730994505789180971543210985669540088948251480817
675264879954136786967328399292610731395201725823776142098567278594552620919467305935173899522941298667043410839269444955
84215958472944393123
C: 901857421401009352117314080877294551717694888475755975119907071775249401691416230597369434433456348351398490878996395
347296015761162362862591020527428935946975980648693351946008209345580514465731887535972736191091277614736858076391792668
323255630109510105302972943821911643125525291471473772318815590251995849464611105993541058726658005560932397610132750877
939117362375852014541072616724056815525363223244932454305420695496211178176310423755815770477439546857670408295123779369
253700077125458156350870569979322443311339207398963180241922468307804231128500989403385677482561657441628233364539602121
4664439255174831040
Start: bn_t = 0, bn_s = 0 : 82
Generated: bn_t = 0, bn_s = 0 : 398159551
MitM message: 551361
MitM message in hex : 869c1
Optimized 'Meet in the middle' execution time: 8.11774e+08 mu s
```

Також ми зробили тест bruteforce, який повідомляє час на 1 спробу підбору.

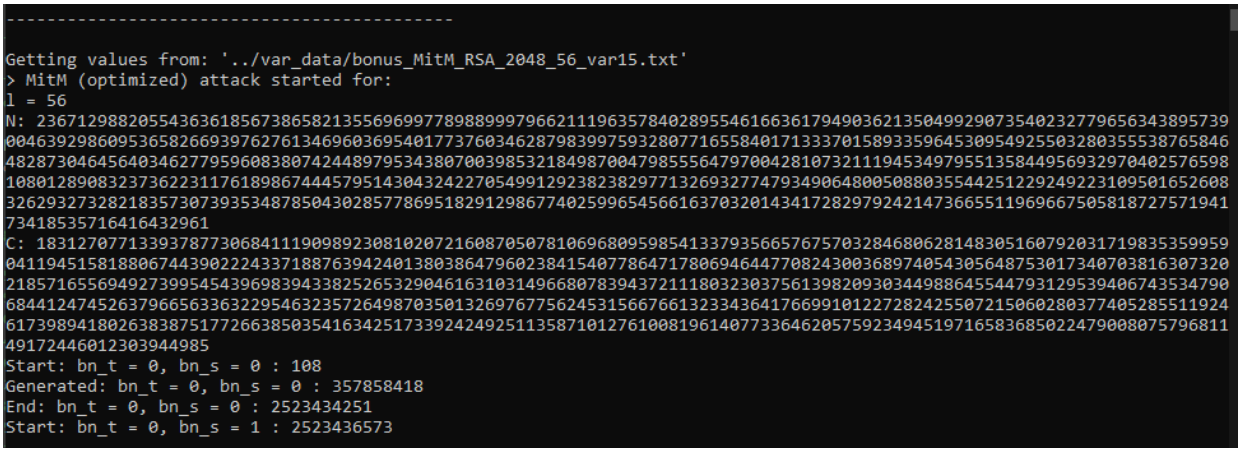
```
-------------------------------------------

Getting values from: '../var_data/MitM_RSA_2048_20_regular_hard_var15.txt'
Bruteforce once try (find next M & get C & make a compare) time: 62.6 mu s
```

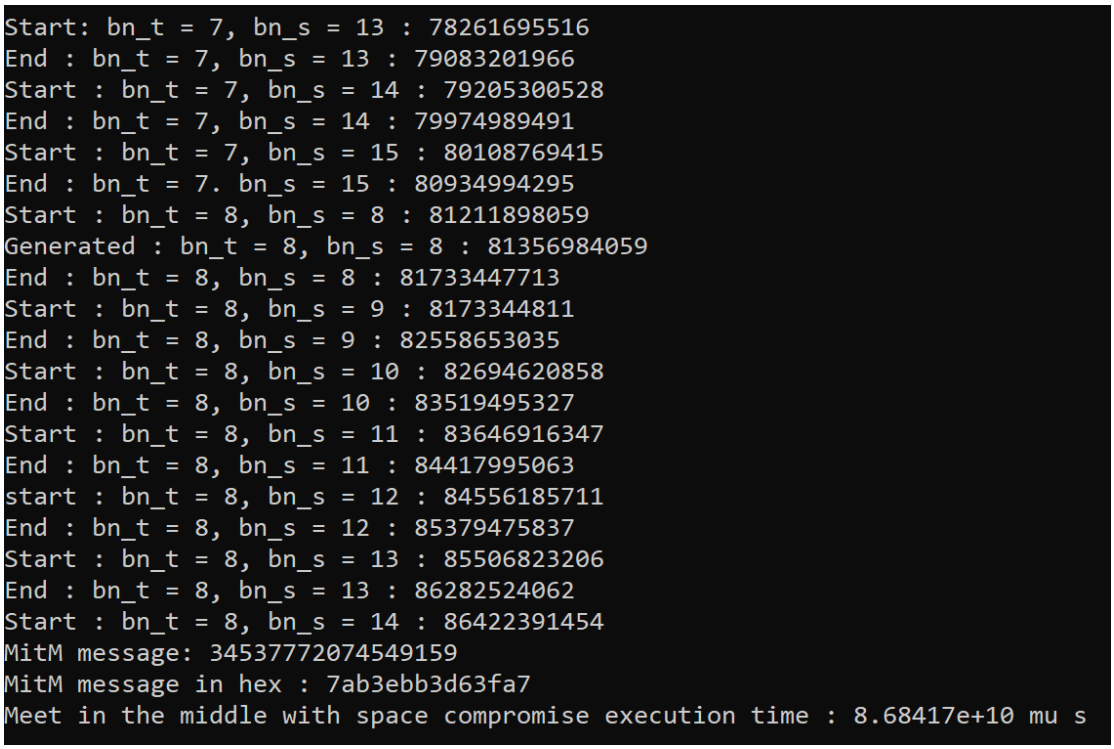
Помноживши кількість можливих варіантів ключів 2^{2048} на $62.6\,\mu s = 62.6 \times 10^{-6}\,s$ маємо, що час на злам прямим перебором займатиме:

$$T \approx 2.02 \times 10^{612}\,seconds \approx 6.41 \times 10^{604}\,year$$

Ця цифра є фактично нескінченністю (порівняно з часом життя Всесвіту який дорівнює $1.3798 \times 10^{10}\,year$) для будь-яких практичних задач. Навіть за умови надзвичайно оптимізованого паралелізму (використання мільярдів ядерів) brutforce 2048-бітного ключа RSA є абсолютно нездійсненним.

Далі ми спробували (і змогли!) зробити бонусне завдання додавши оптимізацію. Його результат є наступним:





Процес зайняв порядку 24.5 годин.

1.4 Висновки:

Завдяки комп'ютерному практикуму ми розібралися з двома атаками на криптосистему RSA, "побачили руками, очами потрогали" як вони реалізовується на практиці. Було важко розібратися з оптимізацією алгоритму-атаки MitM. За допомогою документації C++ ми змогли реалізувати оптимізацію, яка економить ресурси комп'ютера. Довелося робити trade-off крок в бік оптимізації під простір і розбити простір перебору на блоки, генерувати їх, очищати, робити перебір і генерувати знову.

Також були думки серіалізувати список T, $T^e \mod N$, а також були думки з вивантаженням його в постійну пам'ять, але нас зупинили думки про I/O overhead та складна програмна реалізація відповідно. Також була ідея з використанням іншого API arrayfire для пришвидшення роботи алгоритму.

P.S. На жаль, найбільш наївне рішення проблеми не спрацювало :(
https://downloadmoreram.com/