Network Security - Project

Tian Dong tian.dong@sjtu.edu.cn

2023.5

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Contents

Contents

Introduction

- DDL: 2025.06.23 23:59
- Submit Format:
  - Project_${Team_name}.{tar.gz/rar/zip}
    - Task 1
      - Your code/data for task 1 (15/100)
    - Task 2
      - Your code/data for task 2 (25/100)
    - Task 3
      - Your code/data for task 3 (15/100)
    - Report_${Team_name}.pdf (45/100)

Contents

Project – Task 1

- Goal: Implement the textbook RSA algorithm (without any padding)
- Your code should be able to:
  - **Generate** a random RSA key pair with a given key size (e.g., 1024-bit)
  - **Encrypt** a plaintext with the public key.
  - **Decrypt** a ciphertext with the private key.

Contents

Project - Task 2

- Goal : Perform a CCA2 attack on textbook RSA
- Textbook RSA is elegant, but has **no semantic security**.
- An adaptive chosen-ciphertext attack (abbreviated as **CCA2**) is an interactive form of chosen-ciphertext attack in which an attacker sends a number of ciphertexts to be decrypted, then uses the results of these decryptions to select subsequent ciphertexts.
- The goal of this attack is **to gradually reveal** information about an encrypted message, or about the decryption key itself.

Project - Task 2

- Refer an existing work for the implementation
  - Details of this attack can be found in **Chap 4**.

When Textbook RSA is Used to Protect the Privacy of Hundreds of Millions of Users

Jeffrey Knockel
Dept. of Computer Science
University of New Mexico
jeffk@cs.unm.edu

Thomas Ristenpart
Cornell Tech
ristenpart@cornell.edu

Jedidiah R. Crandall
Dept. of Computer Science
University of New Mexico
crandall@cs.unm.edu

- *Knockel J, Ristenpart T, Crandall J. When textbook RSA is used to protect the privacy of hundreds of millions of users[J]. arXiv preprint arXiv:1802.03367, 2018.*

Project - Task 2

Server-client communication

Client

① generate a 128-bit AES session key for the session.

② encrypt this session key using a 1024-bit RSA public key.

③ use the AES session key to encrypt the WUP request.

④ send the RSA-encrypted AES session key and the encrypted WUP request to the server.

---

① decrypt the RSA-encrypted AES key it received from the client.

② choose the least significant 128 bits of the plaintext to be the AES session key.

③ decrypt the WUP request using the AES session key.

④ send an AES-encrypted response if the WUP request is valid.

Server

Project - Task 2

- In this attack, the server knows
  - RSA key pair
  - AES key
- The adversary knows
  - RSA public key
  - a RSA-encrypted AES key
  - an AES-encrypted WUP request
- The adversary wants to know
  - **AES key**

Project - Task 2

- In this part, you are supposed to
  - Properly design your own WUP request format, **server-client communication model**, etc.
  - **Generate** a history message by yourself, it should includes a RSA-encrypted AES key and an AES-encrypted request.
  - Present the **attack** process to obtain the AES key (and further decrypt the encrypted request) from the history message.
- You can use third-party library to implement **AES** encryption and decryption.

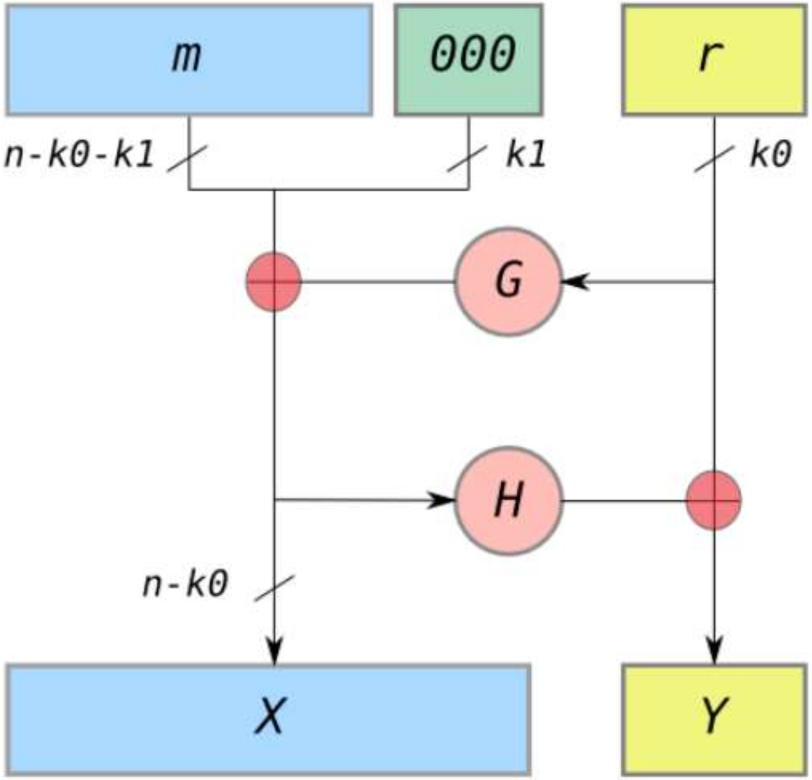
Contents

Project - Task 3

- Goal: defend the attack
  - **Implement RSA-OAEP algorithm and discuss why it can defend such kind of attacks.**
- Since textbook RSA is vulnerable to attacks, in this paper, the authors give a solution: **using OAEP key padding algorithm**.
- In cryptography, Optimal Asymmetric Encryption Padding (**OAEP**) is a padding scheme often used together with RSA encryption. OAEP satisfies the following two goals:
  - Add an element of randomness which can be used to convert a **deterministic** encryption scheme (e.g., traditional RSA) into a **probabilistic** scheme.
  - **Prevent partial decryption** of ciphertexts (or other information leakage) by ensuring that an adversary cannot recover any portion of the plaintext without being able to invert the trapdoor one-way permutation.

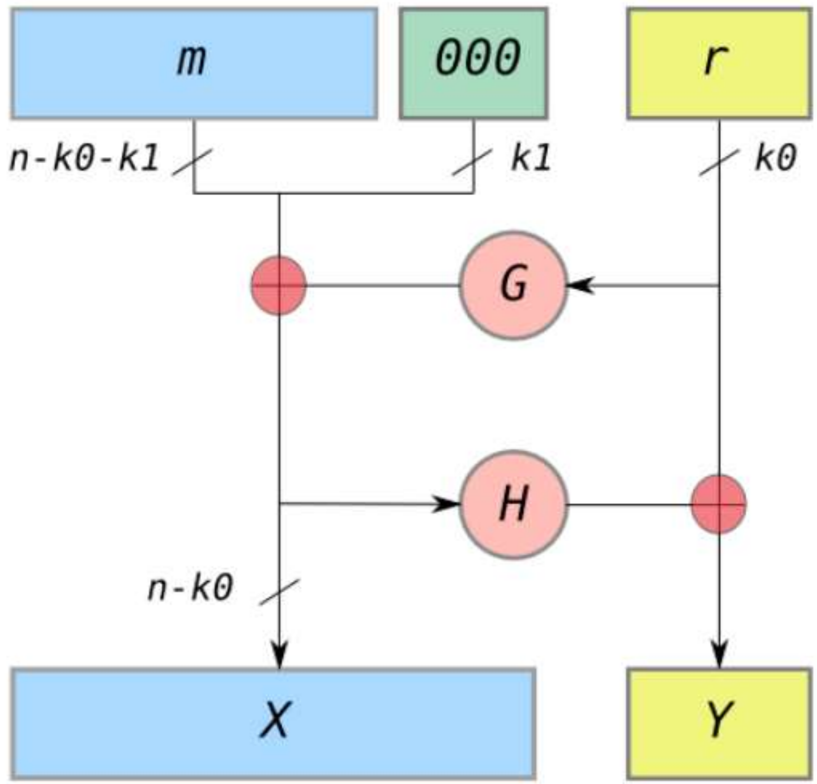
Project - Task 3: OAEP

- *n* is the number of bits in the RSA modulus.
- *k0* and *k1* are integers fixed by the protocol.
- *m* is the plaintext message, an (*n*−*k0*−*k1*) bit string
- *G* and *H* are typically some cryptographic hash functions fixed by the protocol.
- ⊕ is an xor operation



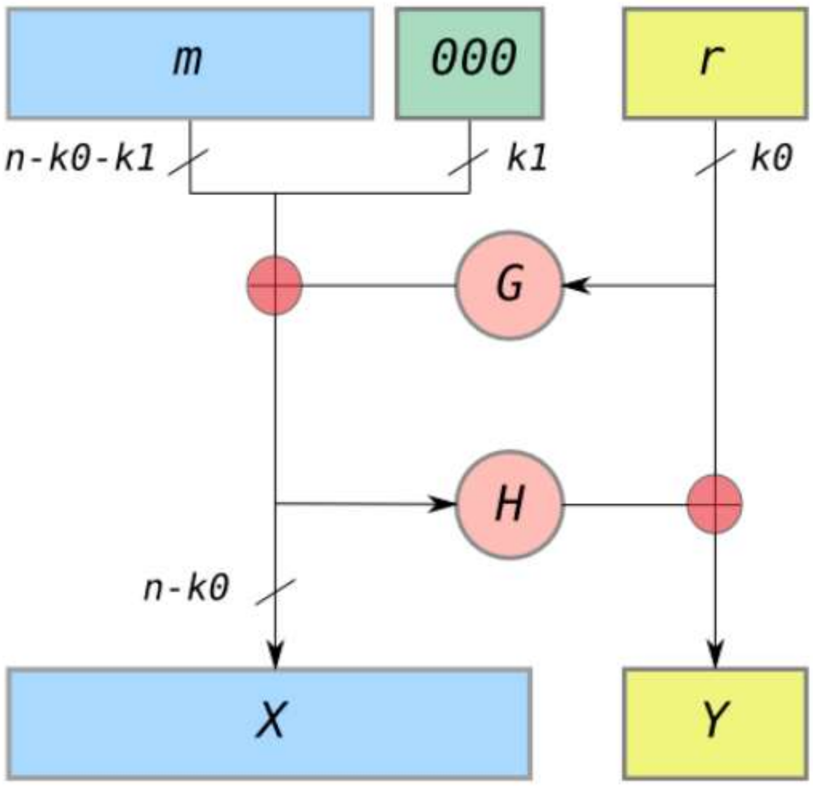
Project - Task 3: OAEP encoding

1. messages are padded with *k1* zeros to be *n*−*k0* bits in length.
2. *r* is a randomly generated *k0* bit string
3. G expands the *k0* bits of *r* to *n*−*k0* bits.
4. X = *m*00..0 ⊕ G(*r*)
5. H reduces the *n*−*k0* bits of X to *k0* bits.
6. Y = *r* ⊕ H(X)
7. The output is X || Y where X is shown in the diagram as the leftmost block and Y as the rightmost block



Project - Task 3: OAEP decoding

1. recover the random string as $r = Y \oplus H(X)$
2. recover the message as $m00..0 = X \oplus G(r)$
3. The "**all-or-nothing**" security is from the fact that to recover *m*, you must recover the entire X and the entire Y; X is required to recover *r* from Y, and *r* is required to recover *m* from X. Since any changed bit of a cryptographic hash completely changes the result, the entire X and the entire Y must both be completely recovered.



Project - Task 3

- In this part, you are supposed to
  - Add the **OAEP padding** module to the textbook RSA implementation.
  - Give a **discussion** on the advantages of RSA-OAEP compared to the textbook RSA.
  - As a bonus, you can further try to **present** CCA2 attack to **RSA-OAEP** to see whether it can thwart the CCA2 attack you have implemented in part 2.

Contents

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Task 1

- Files to be Submitted and Standard of Grading:
  - **Code** : 6 points
  - **RSA parameters (Decimal, 1024bits):**
    - RSA_Moduler.txt 1 point
    - RSA_p.txt 1 point
    - RSA_q.txt 1 point
  - **RSA key (Decimal, 1024bits):**
    - RSA_Secret_Key.txt 1 point
    - RSA_Public_Key.txt 1 point
  - **Encryption:**
    - Raw_Message.txt 1 point
    - Encrypted_Message.txt (hexadecimal) 1 point
    - Pass Decryption (TA) 2 points

Task 2

- Files to be Submitted and Standard of Grading:
  - **Code** : 10 points
  - **CCA2 (Use RSA parameters in task 1):**
    - History_Message.txt 1 point
    - AES_Key.txt (hexadecimal, 128bits) 1 point
    - WUP_Request.txt (hexadecimal) 1 point
    - AES_Encrypted_WUP.txt (hexadecimal) 2 points
    - Attack Process to Obtain the AES key: 10 points
      - Both Screenshot and Log Files are OK

Task 3

- Files to be Submitted and Standard of Grading:
  - **Code** : 10 points
  - **Encryption (Use RSA parameters and Message in task 1):**
    - Random_Number.txt 1 point
    - Message_After_Padding.txt (hexadecimal) 1 point
    - Encrypted_Message.txt (hexadecimal) 1 point
    - Pass Decryption (TA) 2 points

    (Recommended using n=1024, k0=512, hash: sha512)

Any extra file added is OK but need to be explained in report!

Thank You