Network Security - Project

孙随彬 sun1998@sjtu.edu.cn

2021.6.1

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Contents

Contents

Introduction

- Select **one** of two projects, and complete it by **yourself**
- Due date: Friday of 18th week
- Deliverable:
  - Project{1/2}_${Student_id}_${your_name}.{tar.gz/rar/zip}
    - Task 1
      - Your code/data for task 1
    - Task 2
      - Your code/data for task 2
    - Task 3
      - Your code/data for task 3
    - Project{1/2}_report_${Student_id}_${your_name}.pdf

Contents

Project 1 – 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., 1024bit)
  - **Encrypt** a plaintext with the public key.
  - **Decrypt** a ciphertext with the private key.

Project 1 - 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 1 - Task 2

- Refer an existing work for the implementation
  - Details of this attack can be found in **Chap 4**.
- Since QQ browser has fixed the problem, you are supposed to simulate the attack

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 1 - 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 1 - 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 1 - Task 2

- In this part, you are supposed to
  - Properly design your own **WUP** request format, server-client communication model, etc. A nice design will bring you a bonus.
  - **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.

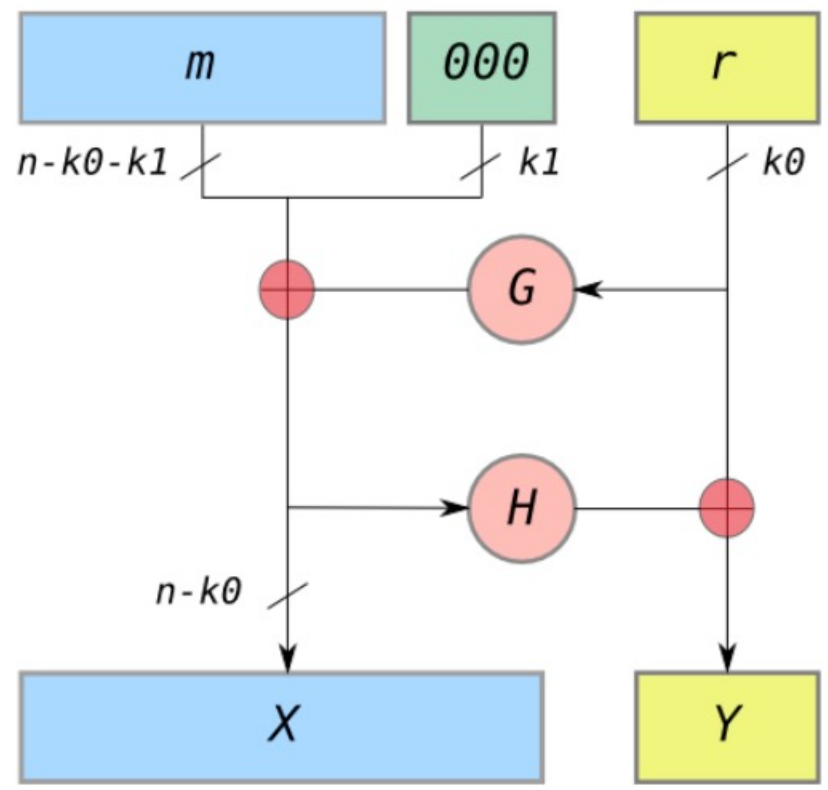
- About WUP: *https://citizenlab.ca/2016/03/privacy-security-issues-qq-browser/*

Project 1 - 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.

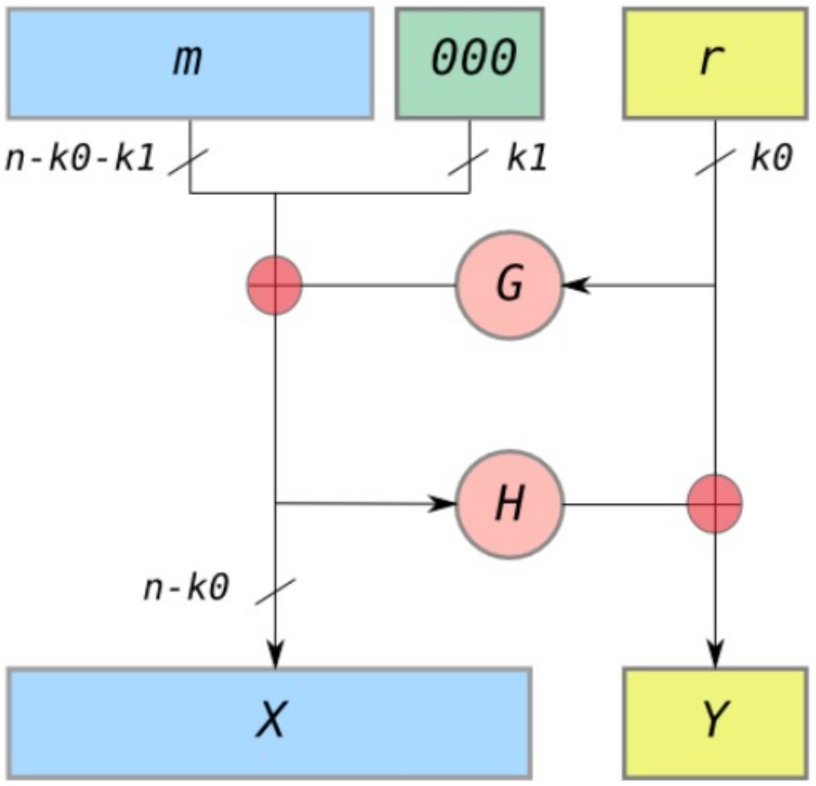
Project 1 - Task 3: OAEP

- 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.
- $\oplus$ is an xor operation



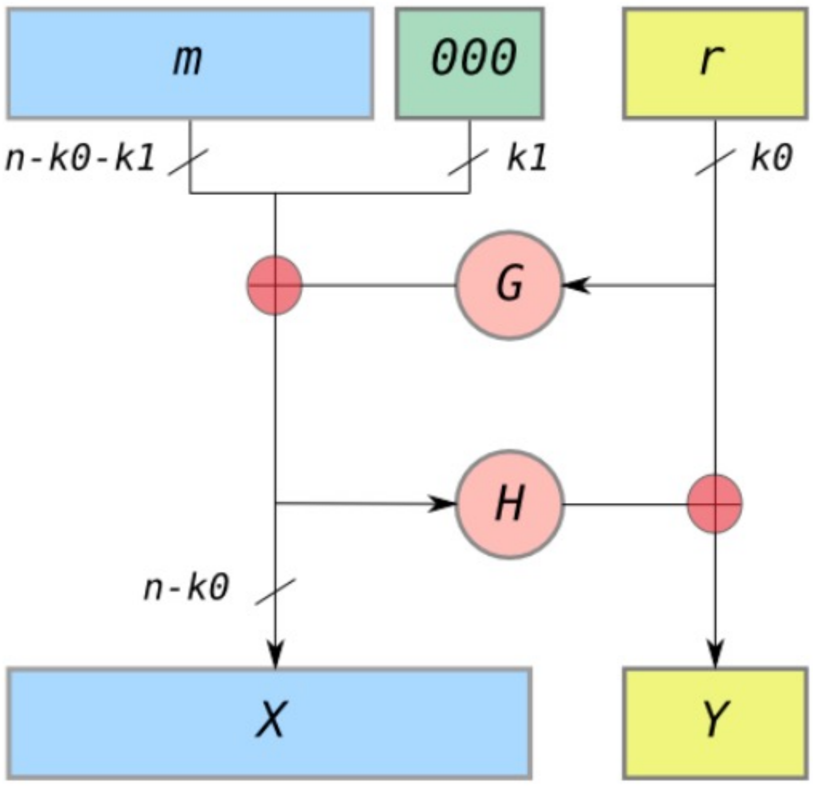
Project 1 - Task 3: OAEP encoding

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. $X = m00..0 \oplus G(r)$
5. *H reduces the n−k0 bits of X to k0 bits.*
6. $Y = r \oplus H(X)$
7. *The output is X || Y where X is shown in the diagram as the leftmost block and Y as the rightmost block*



Project 1 - 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 1 - 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.

Thank You