

Problem 1-1. Weaknesses of CPA-secure cryptosystems

Let $F: \mathcal{K} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a secure pseudorandom function. In class, we saw the CPA-secure encryption scheme (Enc, Dec) with keyspace $\mathcal{K}$, where

$$\text{Enc}(k, m) := \begin{cases} r \xleftarrow{\mathcal{R}} \{0, 1\}^n \\ c \leftarrow F(k, r) \oplus m \\ \text{output } (r, c) \end{cases} \quad \text{Dec}(k, (r, c)) := \begin{cases} m \leftarrow F(k, r) \oplus c \\ \text{output } m \end{cases}.$$

We will use a secure MAC scheme $\text{MAC}: \mathcal{K} \times \{0, 1\}^* \rightarrow \{0, 1\}^n$ over the same keyspace $\mathcal{K}$. MIT uses encryption to protect communications between department offices and its central payroll system. In particular, the EECS department office and MIT's payroll office share a secret encryption key $k_{\text{Enc}} \xleftarrow{\mathcal{R}} \mathcal{K}$ and a secret MAC key $k_{\text{MAC}} \xleftarrow{\mathcal{R}} \mathcal{K}$.

On each pay date, a machine in the EECS department sends a sequence of messages to the payroll office. Each plaintext message is a 54-byte ASCII-encoded string with format:

$m = \text{"date:NNNN-NN-NN, mit_id:NNNNNNNNN, amount:NNNNNNNNN.NN"}$,

where each N represents an ASCII-encoded decimal digit (i.e., ASCII values $0 \times 30 - 0 \times 39$). Unfortunately, the designers of the system make the wrong decision and use MAC-then-encrypt rather than encrypt-then-MAC. In particular, the EECS department encrypts each message m as:

$$\text{ct} = \text{Enc}(k_{\text{Enc}}, m \parallel \text{MAC}(k_{\text{MAC}}, m)).$$

The machine at the payroll office operates on each ciphertext ct as follows:

- Compute $(m' \parallel t') \leftarrow \text{Dec}(k_{\text{Enc}}, \text{ct})$.
- Check that the bytes of the string corresponding to the MIT ID number and the payment amount are valid ASCII decimal digits. If not, send a `FORMAT ERROR` message to the EECS machine.
- Check that the date corresponds to today's date. If not, send a `DATE ERROR` message to the EECS machine.
- Check that $t' = \text{MAC}(k_{\text{MAC}}, m')$. If not, send a `MAC ERROR` message to the EECS machine.
- Append the $(\text{ID}, \text{amount})$ pair to a log file for later processing and send an `OK` message to the EECS machine.

Explain how a network attacker that can intercept ciphertexts and interact with the payroll server can recover **all but one of the bits in each secret digit** in the plaintexts that the EECS department sends. Recovering these bits for a single plaintext using your attack should take no more than 10,000 interactions with the payroll server.

Solution: The attacker intercepts a ciphertext ct and recovers bits from the plaintext one unknown decimal digit at a time. (The attacker knows the date, since it is just the current date.) If we view each decimal digit as a 4-bit number, the attacker will be able to recover all but the least significant bits of the number.

For a number $i \in \{0, \dots, 9\}$, let $\text{ASCII}(i)$ be the ASCII encoding of i as a byte. For a byte b $\text{IS_ASCII}(b)$ be the function that returns “1” if the byte is a valid ASCII-encoded decimal digit and that returns “0” otherwise. For a number $i \in \{0, \dots, 9\}$, define the *fingerprint* of i to be the 16 bits:

$$(\text{IS_ASCII}(\text{ASCII}(i) \oplus 0x00), \dots, \text{IS_ASCII}(\text{ASCII}(i) \oplus 0x0F)).$$

You can convince yourself (e.g., with a short Python program) that the only numbers in $\{0, \dots, 9\}$ that have the same fingerprint are ones that are the same in their most-significant 3 bits. So 4 and 5 have the same fingerprint, but 4 and 7 do not. So, knowing the fingerprint of a number in $\{0, \dots, 9\}$ is enough to determine the number’s most significant three bits.

Next, observe that the attacker in our setting can compute the fingerprint of any of the unknown digits in the plaintext. To compute the fingerprint of the integer at the b th position in the plaintext, the attacker runs:

- For $\sigma \in \{0x00, \dots, 0x0F\}$:
 - Set $ct' \leftarrow ct$ and then $ct'[b] \leftarrow ct[b] \oplus \sigma$.
 - Send ct' to the payroll server.
 - If the server replies OK or MAC ERROR, set $f_\sigma \leftarrow 1$, otherwise set $f_\sigma \leftarrow 0$.
- Return $(f_{0x00}, \dots, f_{0x0F})$ as the fingerprint of the b th digit.

Finally, the attacker can put these ideas to recover the most-significant bits of each unknown message digit. For each unknown decimal digit in the plaintext, the attacker computes the fingerprint. The fingerprint then immediately gives the attacker the three most-significant bits of the unknown digit.

To compute the fingerprint, the attacker generates at most 16 ciphertexts per digit and there 20 unknown digits in the plaintext. So the attack requires at most 320 interactions with the payroll server. With a more careful fingerprinting technique, you can reduce the attack cost further.

Problem 1-2. Encryption and RSA

- (a) Which of the following security goal(s) does encryption address :
(1) Confidentiality (2) Integrity (3) Sender authentication (4) Non-repudiation.

Solution: Encryption aims to provide confidentiality only. Non-repudiation is the concept of ensuring that a party in a dispute cannot repudiate, or refute the validity of a statement or contract.

- (b) Suppose you obtain two ciphertexts C, C' encrypted using one-time pad, with key K and its bitwise complement $\bar{K} = K \oplus 1$ respectively. What can you infer about the corresponding plaintext messages?

Solution: You can obtain $m \oplus m$. Note that $K = K1$, therefore, $C \oplus C \oplus 1 = (m \oplus K) \oplus (m \oplus K \oplus 1) \oplus (1) = m \oplus m$.

- (c) Alice and Bob are using public keys $(e_1, N_1), (e_2, N_2)$ respectively. Suppose you are informed that their RSA moduli N_1, N_2 are not relatively prime. How would you break the security of their subsequent communication? It is sufficient to show that you can get $\phi(N_1)$ and $\phi(N_2)$.

Hint: Euclid's algorithm allows us to compute $\text{GCD}(x,y)$ efficiently.

Solution: Since N_1, N_2 share a common divisor (they are not relatively prime), let p be the shared prime divisor. Let $N_1 = p \cdot q$ and $N_2 = p \cdot r$. $p = \text{GCD}(N_1, N_2)$, which can be computed efficiently using Euclid's Algorithm. We do not expect the solution to outline the algorithm. $q = N_1/p$ and $r = N_2/p$ can be computed with the knowledge of p . $\phi(N_1) = (p-1)(q-1)$ and $\phi(N_2) = (p-1)(r-1)$ can be computed with the knowledge of p, q, r .

- (d) Suppose that a system uses textbook RSA encryption. An attacker wants to decrypt a ciphertext c to obtain the corresponding confidential plaintext m . Assume that the victim system readily decrypts arbitrary ciphertexts that the attacker can choose, except for ciphertext c itself. Show that the attacker can obtain m from c even under this setting, i.e RSA is not CCA secure.

Solution: We know that $c = m^e \pmod{N}$, i.e, the attacker aims to obtain m from c . To do this, the attacker uses the following scheme.

- Step 1. The attacker chooses a random number r' such that $\text{gcd}(r', N) = 1$. The attacker encrypts it as using the public key : $C_r = r'^e \pmod{N}$.
- Step 2.
He computes $C = c \cdot C_r \pmod{N}$, and asks the system to decrypt C

Let the decryption be M .

By definition of RSA encryption, we know that: $M' = (C')^d \bmod N$.

It follows that: $C' = ((m^e \bmod N) \cdot (r^e \bmod N)) \bmod N = (m \cdot r)^e \bmod N$

Therefore, $M' = C'^d \bmod N, M' = (m \cdot r)^{ed} \bmod N$

This follows from Fermat's little theorem and the construction of e, d .

$M = (m \cdot r) \bmod N$

- Step 3. The attacker obtains M and recovers the confidential plaintext m by computing $M \cdot r^{-1} \bmod N$.
 $M \cdot r^{-1} \bmod N = m \cdot r \cdot r^{-1} \bmod N = m \bmod N$.
- Keep in mind that r has to be chosen such that its multiplicative inverse modulo N exists. This is true iff $\gcd(r, N) = 1$.