CS6111: Foundations of Cryptography

Assignment 5

S Srinivas Saurab
CS16B039

Instructions

- Deadline: 25 Nov 2019 (23:59)

1. (6 points) **RSA Encryption** $pk = (n, e)$ where $n = pq$ and p, q are two large primes. $sk = (n, d)$ where $ed \equiv 1 \mod \phi(n)$. In "textbook RSA", $Enc(pk, m) = m^e \mod n$ and $Dec(sk, c) = c^d \mod n$.

   (a) (2 points) Explain why the "textbook RSA" scheme is not CPA-secure.

   Solution: Textbook RSA is not CPA secure because it is deterministic. There is nothing preventing the Adversary with access to Encryption Oracle to computer the cipher-texts for m_o and m_1 in the security game and compare them with the given text $Enc(m_b)$. This way, as long as $Enc(m_o)$ and $Enc(m_1)$ are distinct, the adversary can always succeed. This is always the case because the decryption must be uniquely determinable from the cipher-text.

   (b) (2 points) Modify the RSA protocol such that it achieves CPA security. Give just the idea. Proof is not required.

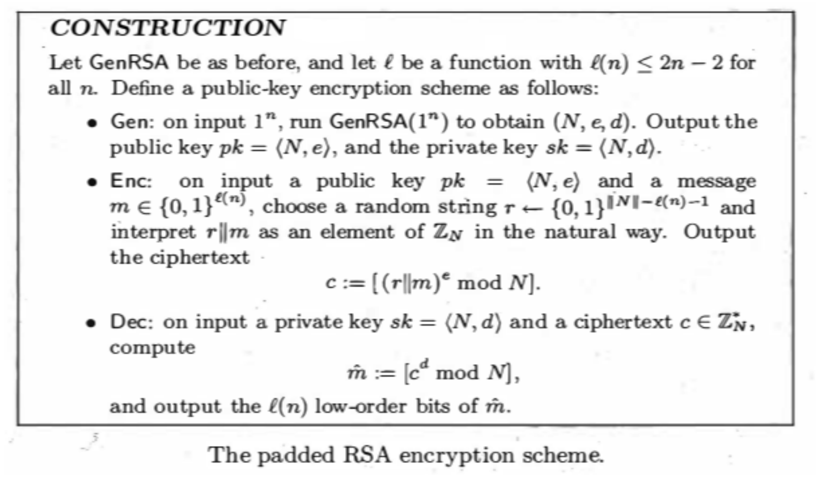
   Solution: A simple fix to make "textbook RSA" CPA secure is be adding a random pad to each message before encryption. This way it becomes non-deterministic. Moreover, this random pad can then be removed on the receiving end by considering the final few bits alone. See Figure-1 for exact details of the construction.

   Consider the "textbook" (hashless) RSA-signature scheme. Given a message m, let $\sigma(m) = m^d \mod n$. Verification checks if $\sigma^e = m$.

   (c) (2 points) Show that this signature scheme is *universally* forgeable (you can obtain the signature of any new message of your choice) under a chosen message attack.

   Solution: In chosen message attack, we can query the encryption oracle for any encryption other than m itself. Since 'm' is an element in additive group of n=pq, let us consider the element m(n-1). m(n-1) mod N$\in \mathbb{Z}_n$. Hence it would be valid to query for its encryption. Let 's' be the sign which is output by Enc Oracle:

   $$s = (m(n-1))^d (mod \;\; n)$$



CONSTRUCTION

Let GenRSA be as before, and let ℓ be a function with $\ell(n) \leq 2n-2$ for all n. Define a public-key encryption scheme as follows:

- Gen: on input 1^n, run GenRSA(1^n) to obtain (N, e, d). Output the public key $pk = \langle N, e\rangle$, and the private key $sk = \langle N, d\rangle$.
- Enc: on input a public key $pk = \langle N, e\rangle$ and a message $m \in \{0,1\}^{\ell(n)}$, choose a random string $r \leftarrow \{0,1\}^{\|N\|-\ell(n)-1}$ and interpret $r\|m$ as an element of $\mathbb{Z}_N$ in the natural way. Output the ciphertext
$$c := [(r\|m)^e \bmod N].$$
- Dec: on input a private key $sk = \langle N, d\rangle$ and a ciphertext $c \in \mathbb{Z}_N^*$, compute
$$\hat{m} := [c^d \bmod N],$$
and output the $\ell(n)$ low-order bits of $\hat{m}$.

The padded RSA encryption scheme.

Figure 1: Padded RSA construction for Q1.b

$$s = m^d(n-1)^d(mod\ \ n)$$

We know that $ed \equiv 1 \mod \phi(n)$. Therefore, d must be even. Hence, we have

$$s = m^d(mod\ \ n)$$

From the last equation, we see that we have produced a forgery for any message 'm'. In the case m(n-1) = m (mod n), we have m = 0, for which we can return zero directly. Thereby, without querying the signature of a message directly, we can produce a forgery for any given message.

2. (11 points) **Hash and CCA** Suppose (Enc, Dec) is a CPA-secure encryption scheme. We shall write $Enc_{pk}(m;r)$ to indicate encryption of a message m using randomness r, where Enc requires $r \leftarrow \{0,1\}^k$ and k is determined by the security parameter. H is a hash function modelled as a random oracle with k-bit output.

   Consider a new encryption scheme with the encryption algorithm defined as follows:

$$Enc^*_{pk}(m;r) = (Enc_{pk}(m\|r;H(r))\ ,\ H(m\|r)),\ \text{where } r \in \{0,1\}^k.$$

   (a) (2 points) What is the corresponding decryption algorithm Dec^*? For a scheme to be CCA-secure, the decryption algorithm must verify some condition and reject all inputs that fail.

Solution:

Algorithm for Dec^*

Input: $< c_1, c_2 >$

Output: $Dec^*(< c_1, c_2 >)$

Algorithm:

$$Dec^*(< c_1, c_2 >) = \begin{cases} m, & \text{if } \exists y, Dec(c_1) = y \text{ and } c_2 = H(y), \text{ where } y = m\|r \\ \perp, & \text{if Dec}(c_1) = \perp \\ \perp, & \text{otherwise} \end{cases}$$

Brief Proof of CCA-security:
Since Enc,Dec is CPA-secure, we first prove Enc*,Dec* if CPA-secure. This is the case because, if it was not, we can construct a CPA-adversarial example for Enc,Dec scheme. From here, we argue that it is indeed CCA because the random oracle based Dec-oracle offers virtually no extra information over the information gained in the Encryption phase. This is due to tactical $\perp$ returns on any ciphertext modification.

(b) (2 points) Show that the scheme will not be CPA-secure if $H(m\|r)$ is replaced with $H(m)$. (The answer is obvious, but since this is the random oracle model, it requires two encryption queries first. Write them.).

Solution: The second part of the input will become deterministic as the Hash function is computed with no randomness in $H(m)$. The following Adversary A will break the CPA-security:
Algorithm for A:

```
    Choose fixed $m_o$ and $m_1$
    Query random oracle emulating H for $s_o = H(m_o)$ and $s_1 = H(m_1)$
    $return \begin{cases} 1, & \text{if } s_1 = c_2 \\ 0, & \text{if } s_o = c_2 \end{cases}$
```

where $< c_1, c_2 >$ is the ciphertext in A's hands. CPA-secure is broken because of determinism in c_2.

We will prove that (Enc^*, Dec^*) is indeed a CCA-secure encryption scheme in the random oracle model. Fill in the following reduction from the CPA-security of Enc to the CCA-security of Enc^*. Let A^* be a CCA-adversary for (Enc^*, Dec^*). We will obtain a CPA-adversary A for Enc as follows. That is:

- A interacts with the CPA-challenger C for Enc and is given public key pk
- A simulates the CCA-challenger for Enc^* to interact with blackbox A^*

Solution: The aim for A here is to perfectly simulate Enc*, Dec* and tactfully use this simulation to break the CPA-security of Enc,Dec. Let us look at three behaviours of A on A*'s - Enc query, Dec query, message pair selection (m_o, m_1). Moreover, A makes the Hash table with (input,output) entries corresponding to all hash computations throughout it's interaction with A*. Let us call it $\mathbb{T}$. It also stores all the Encryptions and Decryptions performed by it as a proxy for (Enc*,Dec*) scheme in the table $\mathbb{ED}$.

(c) (2 points) How are the encryption queries of A^* answered? Give format of query and reply.

Solution: We can look at A to be emulating like A*'s challenger. Whenever A* makes a query for message 'm', A appends a random value 'r' to it and gets r||m. Then, it finds H(r) and H(m||r) to get find the Enc* directly using Enc oracle of it's own CPA game. Finally it computes Enc* with all these components and returns it to A*. This is a perfect proxy.

Also, A adds the pairs (r,H(r)) and (m||r, H(m||r)) to $\mathbb{T}$. It also adds <m||r, Enc*(m||r)> to the table $\mathbb{ED}$. Note that 'r' must be chosen in such a way as to avoid collision with entries in table $\mathbb{T}$.

(d) (2 points) How are the decryption queries of A^* answered? Give format of query and reply.

Solution:
Input: $< c_1, c_2 >$
Output: m
Decryption: A can just use the table $\mathbb{ED}$ to provide m, if $(m, < c_1, c_2 >)$ actually exists in this table. If no match for $< c_1, c_2 >$ exists in the table, return $\perp$.

(e) (3 points) Once A^* provides its challenge messages $\{m_0, m_1\}$, what does A do? (Note that A does not know the randomness C uses to encrypt m_b. Does this matter?)

Solution: A* must tactically use this opportunity to break out of its CPA-security. If A receives m_o, m_1, it can send the same messages to its challenger C. Upon receiving $Enc_{pk}(m_b \| r)$, it tosses a coin by itself to get b'. Now, it computes $H(m'_b \| r')$ where r' is a new random value chosen so as to avoid collisions.

A can now return whatever A* does and be successful with the same non-negligible probability. This is because, the computation $H(m'_b \| r')$ can be taken as the actual random output of the random oracle emulating H. Therefore, it would not make a difference to the advantage of A*. Therefore, A* would still succeed with non-negl advantage which would be transferred to A that merely copies A*'s output.

Random-oracle model means the following: every hash query of the CCA-adversary is answered by the CCA-challenger. In this example, every hash query x that A^* makes is sent to A, who then answers it by sampling and returning a random output y and tabulates $H(x) = y$. Thus, A knows the inverse of every hash value A^* uses.

Ensure that you understand how to draw and describe the full reduction à la the nested encryption question from the Midsem.

3. (8 points) **Insecure Signatures** Consider the following signature scheme: the public key of this signature scheme consists of a Blum integer $N = pq$ (Blum integer is when -1 is a non-square mod p and mod q). The secret key is p, q. A signature on message m is as follows:

- If m is quadratic residue, then σ is any square root of m.
- Else, σ is any arbitrary square root of $-m$.

(a) (2 points) Given two square roots x, y of the same value $a \mod n$ such that $x \not\equiv y \mod n$, show that one can factor n.

Solution: Given x,y which follow $x^2 \equiv c$ mod n and $y^2 \equiv c$ mod n, we can infer that

$$n | x^2 - y^2$$

$$n | (x+y)(x-y)$$

From the question, we know that $n \nmid x-y$ or $x+y$. Hence, $pq|(x+y)(x-y)$ implies one of x+y and x-y is p and the other is q. Two compute p,q exactly, we can use Euclid's GCD algorithm for gcd(n,x-y) which can be computed by deterministic PT algorithms. This way, one can factor n, if such x,y are given.

Recall the two types of forgery on signature-schemes: existential forgery and target-message forgery.

(b) (2 points) With a public-key only attack (ie, with access to only the signer's public key, and not the signing algorithm), a target-message attack (ie, producing a signature on messages of one's choice) on this signature scheme is as hard as factoring.

Solution: To show TM-Attack is as hard as factoring, it is enough to show that factoring can be done each time a successful TM-attack is performed.
If $\mathbb{W}$ is the successful TM-attack adversary, we have:
Algorithm for adversary:

```
While (n- -):
    Select y ∈ $\mathbb{Z}_n^*$
    x := $y^2$
    r := W(x)
    If r == $\pm y$
        Continue
    Else
        Factor using Q3.a and return <p,q>
```

The probability that factoring step does not come-by is $\frac{1}{2^n}$, which is negligible under asymptotic considerations.

(c) (2 points) Show that this scheme is existentially forgeable with a public-key only attack.

Solution: With m = 1, we see that $\sigma = 1$. This is independent of <pk,sk> combination. Hence there is a ubiquitous existential forgery for the message '1' across all keys.

(d) (2 points) Show that the secret-key can be recovered by an adversary that can interact with the signer (i.e., make queries to the signing algorithm).

Solution: The solution is similar to Q3.b. Instead of successful TM-attack adversary, we use the signing algorithm itself. Because all that the algorithm needs is the signature for any query message, both offer the same functionality. Therefore, the negligibility proof is intact as well. As the secret key is <p,q> the problem of factoring n=pq in Q3.b exactly matches with the problem of retreiving the secret key (both are completely solved iff p,q pair is found out).