RSA-OAEP

T-79.5502 Advanced Course in Cryptology, Spring 2006

Vesa Vaskelainen

Overview

- Introduction
- The Optimal Asymmetric Encryption Padding
- Random Oracle Model for Security Proof
- RSA-OAEP
- A reduction based on Random Oracle Model

Introduction

- Asymmetric encryption had before Bellare and Rogaway's invention widely-recognized gap between practical schemes and provably-secure ones (IND-CCA2)
- The goal of Bellare and Rogaway was to do asymmetric encryption in a way as efficient as any mechanism at that time suggested and to achieve provable security against IND-CCA2
- In the setup we consider a sender who holds a k-bit to k-bit trapdoor permutation f and wants to transmit a message x to a receiver who holds the inverse permutation f^{-1}.

- Practitioners want that encryption should require just one computation of f and decryption should require just one computation of f^{-1} and also the length of the enciphered text should be precisely k and the length n of the text x that can be encrypted is close to k.
- RSA-OAEP meets the above constraints and it has been widely recognized by the practitioners so that the scheme has been accepted as the RSA encryption standard under industrial and international standardization organizations.
- Next The Optimal Asymmetric Encryption Padding will be described.

Optimal Asymmetric Encryption Padding

- M. Bellare and P. Rogaway. Optimal asymmetric encryption. *Advances in Cryptology - Proceedings of EUROCRYPT'94*.
- randomized message padding technique
- domain of a one-way trapdoor permutation (OWTP)
- The RSA and Rabin functions are the best-known OWTP
- Figure 15.1. We fix the length of the plaintext message as $n = k - k_0 - k_1$ bits, (shorter messages can be suitably padded to this length). The OAEP makes use of two hash functions: a "generator" $G : \{0,1\}^{k_0} \to \{0,1\}^{n+k_1}$ and a "hash function" $H : \{0,1\}^{n+k_1} \to \{0,1\}^{k_0}$.

- An OAEP based public-key encryption scheme can be viewed as a sequentially combined transformation.
- Plaintext $\longmapsto^{OAEP}$ Domain of OWTP $\longmapsto^{OWTP}$ Ciphertext

Central Idea Behind the Transformation

- Mixing of Different Algebraic Structures
- Plaintext Randomization
- Data Integrity Protection, plaintext awareness
- Active attacks are prevented
- If randomized padding output has a good random distribution over the input message space of the OWTP

Random Oracle Model

- A **random oracle** is a powerful and imaginary function which is deterministic and efficient and has uniform output values.
- Bellare and Rogaway's model for security proof is called **random oracle model (ROM)**
- a special agent, **Simon Simulator** is also used
- Simon simulates the behavior of everybody's random oracles

How Simon Simulates

- For oracle G for example, Simon maintains a G-list which contains all the pairs $(a, G(a))$
- for each query a, Simon checks whether or not a is already in the list; if it is, he returns $G(a)$ as the query result; otherwise, he invents a new value for $G(a)$ at uniformly random in the range of G and returns this new value as the query result and archive the new pair $(a, G(a))$ in the list.
- simulated G is deterministic and uniform

- Simon can build the list so that the pairs are sorted by the first element. Then there is no need to apply a sorting algorithm. For each query, a search through a sorted list of N elements can be done in $\log N$ time, in PPT in the size of elements.
- simulated G is efficient

Lemma 15.1: *A random oracle can be simulated perfectly in PPT.*

- Using OWTP-OAEP Simon can simulate random oracles in such a way that he can construct a 1-to-1 relation between plaintexts and ciphertexts.
- If an attacker constructs a *valid* chosen ciphertext using an OWTP f, Simon shall be able to "decrypt" c even though he does not have the trapdoor information for inverting f.
- Simon can also simulate a decryption oracle (e.g., a naive user tricked by Malice to provide a decryption service)
- The simulated "decryption" capability enables Simon to offer a proper "cryptanalysis training course" to Malice in IND-ATK games. (ATK $=$ CPA, CCA or CCA2)

RSA-OAEP

- In the case of **RSA-OAEP**, the OWTP is the RSA encryption function.
- RSA-OAEP encryption scheme involves two hash function evaluations followed by an application of the RSA function, and since hash function can be efficiently evaluated, the scheme is very efficient, almost as efficient as the textbook RSA.
- consider using RSA modulus of the standard length of 2048 bits $(= k)$, and consider $k_0 = k_1 = 160$ (so that 2^{-k_0} and 2^{-k_1} are negligibly small), then the plaintext message encrypted inside the RSA-OAEP scheme can have a lenght up to 84% of the lenght of the modulus.

Towards the Security proof

- If Simon's "training course" is provided at the precise quality then Malice, as a successful attacker, must end up with a non-negligible advantage in short enough time (PPT) to break the encryption scheme (i.e., in the IND-ATK case, he ends up relating one of the two chosen plaintexts to the challenge ciphertext).
- Then Simon who simulates random oracles shall also end up with a successful inversion of the cryptographic function at the point of the challenge ciphertext: the pair (plaintext, challenge ciphertext) can be found in one of his simulated random oracle lists.

- The original ROM-based proof for f-OAEP by Bellare and Rogaway tried to relate an attack on the f-OAEP scheme in the IND-CCA2 mode to the problem of inverting the OWTP f without using the trapdoor information of f.
- However, Shoup has made an ingenious observation and revealed a flaw in that proof.
- Fortunately quickly, the danger of losing a successful public-key encryption algorithm standard was over!
- Fujisaki et al. find a way to rescue OAEP for f being the RSA function.

A Reduction Based on ROM

- Suppose that an attacker $\mathcal{A}$, who is a PPT algorithm, can have a non-negligible advantage to break an f-OAEP scheme in the IND-CCA2 mode. Let us construct an algorithm which will enable Simon Simulator to make use of the IND-CCA2 attacker $\mathcal{A}$ to invert the OWTP f, also with non-negligible advantage.
- This algorithm must be efficient (a PPT one). Thus, Simon efficiently "reduces" his task of inverting f to $\mathcal{A}$'s capability to attacking the f-OAEP scheme.
- The algorithm is therefore called a **polynomial-time reduction**.
- Inversion of f as the combination of $\mathcal{A}$ and the reduction conducted by Simon then runs in polynomial time.

- It is the belief that inversion of f cannot be done in PPT that should refute the existence of the alleged IND-CCA2 attacker $\mathcal{A}$ on f-OAEP.
- The reduction is considered to lead to a contradiction. Therefore, the proof so constructed is called **reduction to contradiction** or **reductionist proof**.
- Let us now study the reduction in detail.

The Reduction

- Let Simon be given the description of an OWTP f and a uniformly random point c^* in the range of f. Simon wants to uncover $f^{-1}(c^*)$ by using $\mathcal{A}$ as an IND-CCA2 attacker.
- Simon has taken over all the communication links of $\mathcal{A}$ to and from the external world
- Simon starts by sending the description of the f-OAEP encryption algorithm to $\mathcal{A}$.
- Simon shall play with $\mathcal{A}$ an IND-CCA2 attack game. Simon shall impersonate the decryption oracle $\mathcal{O}$ as if he has in his possession a valid decryption box. The impersonation is via simulation. We shall see that in ROM, Simon can indeed do so without $\mathcal{A}$ detecting anything wrong.

- Simon shall also provide $\mathcal{A}$ with simulated services for the random oracles G and H used in OAEP. So whenever $\mathcal{A}$ wants to apply G and/or H, it shall actually make queries to Simon and subsequently gets the respective query results back from Simon.
- It is vitally important that the simulations provided by Simon must be accurate so that $\mathcal{A}$ cannot feel anything wrong in its communications with the outside world.
- Only under a precise simulation $\mathcal{A}$ can be educated properly by Simon and thereby release its attacking capacity fully.

The IND-CCA2 attacking game

- Recall **Protocol 14.4**.
- (i) In $\mathcal{A}$'s "find stage", Simon shall receive from $\mathcal{A}$ indifferent chosen-ciphertexts for decryption. $\mathcal{A}$ has freedom to construct these ciphertexts in any way it wishes; but if it does want to construct them properly, via applying the random oracles, then its queries must go to Simon.
- (ii) Since Simon receives from $\mathcal{A}$ chosen ciphertexts for decryption, Simon shall answer them to $\mathcal{A}$ by simulating the decryption box (oracle $\mathcal{O}$).
- (iii) $\mathcal{A}$ shall end its "find stage" by submitting to Simon a pair of chosen plaintexts m_0, m_1. Upon their receipt, Simon shall flip a fair coin $b \in_U \{0,1\}$, and send to $\mathcal{A}$ the "challenge ciphertext" c^* as a simulated f-OAEP encryption of m_b. Here, Simon pretends as if c^* encrypts m_b.

- (iv) Now $\mathcal{A}$ is in its "guess stage". So it may submit further adaptive chosen-ciphertext for its "extended cryptanalysis training course". Simon shall serve as in (ii). In case $\mathcal{A}$ makes random oracle queries in its proper construction of the adaptive chosen-ciphertexts, Simon shall serve as in (i).
- Eventually, $\mathcal{A}$ should output its educated guess on the bit b. This is the end of the attacking game.
- $\mathcal{A}$ should not submit the "challenge ciphertext" c^* for encryption, because then it would be impossible for Simon to provide a simulated decryption since c^* is the very ciphertext that Simon needs $\mathcal{A}$'s help to decrypt.

Simulation of Random Oracles

- In the simulation, Simon maintains two lists, called his G-list and his H-list, both are initially set to empty.
- **G-oracle** Suppose $\mathcal{A}$ makes G-query g. Simon shall first search his G-list trying to find g. If g is found in the list, Simon shall provide $G(g)$ to $\mathcal{A}$. Otherwise, g is fresh; then Simon picks at uniformly random a string $G(g)$ of length $k - k_0$, provides $G(g)$ to $\mathcal{A}$ and adds the new pair $(g, G(g))$ to G-list.
- If the query occurs in $\mathcal{A}$'s "guess stage", then Simon shall try to invert f at the point c^*. He should do for each $(g, G(g)) \in G$-list and each $(h, H(h)) \in H$-list, Simon builds $w = h \parallel (g \oplus H(h))$ and checks whether $c^* = f(w)$. If this holds for some so-constructed string, then $f^{-1}(c^*)$ has been found.

- **H-oracle** Suppose $\mathcal{A}$ makes H-query h. Simon shall first search his H-list trying to find h. If h is found in the list, Simon shall provide $H(h)$ to $\mathcal{A}$. Otherwise, h is fresh; then Simon picks at uniformly random a string $H(h)$ of length k_0, provides $H(h)$ to $\mathcal{A}$ and adds the new pair $(h, H(h))$ to H-list.
- If the query occurs in $\mathcal{A}$'s "guess stage", Simon shall do the same as in the case of G-oracle.
- Notice that if $\mathcal{A}$ has managed somehow to gain some information from c^*, queries in $\mathcal{A}$'s "guess stage" should give that knowledge to Simon and thus help him to decrypt c^*.

Simulation of the Decryption Oracle

- Simon shall simulate the decryption box (oracle $\mathcal{O}$). His simulation steps are: upon receipt of ciphertext c from $\mathcal{A}$ for decryption, Simon looks at each query-answer $(g, G(g)) \in G$-list and $(h, H(h)) \in H$-list; for each pair taken from both lists, Simon computes

$$w = h \parallel (g \oplus H(h)),$$

$$v = G(g) \oplus h,$$

and checks whether

$$c = f(w)?$$

and

Does v have k_1 trailing zeros?

- If both are "YES", Simon shall return the most significant n bits of v to $\mathcal{A}$. Otherwise, Simon shall return REJECT to $\mathcal{A}$.
- $\mathcal{A}$ is polynomially bounded, that is the number of random oracle queries and decryption requests made by $\mathcal{A}$ are also polynomially bounded. Hence, Simon can run the simulated game in polynomial time.

Accuracy of the Simulation

- Let Simon be given a chosen ciphertext c. Simon's simulation for the decryption box is in fact very accurate. Let,

$$s \parallel t = f^{-1}(c),$$

$$r = t \oplus H(s),$$

$$m \parallel 0^{k_1} = s \oplus G(r)$$

  be the values which are defined by c and for which c should be a valid ciphertext.

- If the correct s defined by c has not been queried for random oracle H, then the correct $H(s)$ is missing. In each G-query, we have probability 2^{-k_0} for r being correct. The correct value r is also missing with probability 2^{-k_0}. Value $s \oplus G(r)$ can have probability 2^{-k_1} to have k_1 trailing zeros.

In summary, we can conclude the following result regarding the simulated decryption of a chosen ciphertext c:

- If both s and r have been queried for the respective random oracles, then the simulated decryption can correctly construct $f^{-1}(c)$ and thereby further decrypt c in the usual way.
- If either s and/or r has not been queried for the respective random oracles, then it is correct for the simulated decryption to return REJECT expect for an error probability $2^{-k_0} + 2^{-k_1}$.
- Now we have shown that in the "find stage" the simulated decryption works accurately except for a small error probability. This is enough to show that f-OAEP is provably secure against IND-CCA.

Incompleteness

- Consider the following values defined by c^*

$$s^* \parallel t^* = f^{-1}(c^*),$$

$$r^* = t^* \oplus H(s^*),$$

$$m_b \parallel 0^{k_1} = s^* \oplus G(r^*).$$

- Let us now imagine that s^* is queried for random oracle H. In the "find stage" this must be very unlikely. This is why it was concluded that we already have a valid proof for f-OAEP being secure in the IND-CCA mode. It follows that it is also very unlikely to get m_b in the "find stage" and thus attack against IND-CCA is unsuccessful.
- However, it may be possible in the "guess stage"!

Summary

- Bellare and Rogaway's proof had a flaw
- Can be rescued for f being the RSA function
- In this talk was given a top-level description of the reduction algorithm used in f-OAEP security proof by Bellare and Rogaway and shown that f-OAEP is IND-CCA secure.