Public-key Encryption February 5, 2013

Lecture 12

Lecturer: Dominique Schröder

1 OAEP$^+$

In this lecture, we are proving the security of Shoup's modification of the original OAEP construction that is secure for arbitrary trapdoor permutation. We denote this modified version by OAEP$^+$ and we discuss the construction in the following. Our exposition follows [5] and deviates from the original proof.

1.1 The Scheme

Let $\mathcal{F}$ be a trapdoor one-way permutation, acting over λ bits and let λ_0, λ_1 be two parameters such that $\lambda_0 + \lambda_1 \leq \lambda$ and the functions $2^{-\lambda_0}$ and $2^{-\lambda_1}$ are negligible. The public-key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ encrypts messages $m \in \{0,1\}^n$ with $n = \lambda - \lambda_0 - \lambda_1$ and it uses the following three functions:

$$\begin{aligned} G &: \{0,1\}^{\lambda_0} \mapsto \{0,1\}^n \\ H' &: \{0,1\}^{n+\lambda_0} \mapsto \{0,1\}^{\lambda_1} \\ H &: \{0,1\}^{n+\lambda_1} \mapsto \{0,1\}^{\lambda_0} \end{aligned}$$

In the proof of security, we will model these functions as independent random oraclesl The scheme PKE is defined as follows: The main intuition behind this scheme is similar to the

```
Gen(1^λ)
    $(f, f^{-1}) \leftarrow \mathsf{Gen}_{td}(1^\lambda)$
    $dk \leftarrow f^{-1}, ek \leftarrow f$
    return $(dk, ek)$
```

```
Enc(ek, m)
    parse $m \in \{0,1\}^n$
    $x \leftarrow \{0,1\}^{\lambda_0}$
    $s \leftarrow (G(x) \oplus m) \| H'(x \| m))$
    $t \leftarrow H(s) \oplus x$
    $y \leftarrow f(s \| t)$
    return $y$
```

```
Dec(dk, y)
    $s \| t \leftarrow f^{-1}(y)$
        where $|s| = n + k_1$
    $x \leftarrow H(s) \oplus t$
    parse $s = s_1 \| s_2$
        where $|s_1| = n; |s_2| = \lambda_1$
    $m \leftarrow G(x) \oplus s_1$
    if $H'(x \| m) = s_2$
        return $m$
    else $\perp$
```

previous encryption schemes in the random oracle model: A simulator that does not know the trapdoor can still simulate the decryption query only by observing the queries to the different random oracles.

Theorem 1 *If $\mathcal{F}$ is a trapdoor one-way permutation, then* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is CCA2 in the random oracle model.*

Proof The proof presented in the lecture is different from the one of [4]; our exposition follows [5]. Assume that $\mathcal{A}$ is an attacker against the CCA2 security of OAEP$^+$. This algorithm, has black-box access to the random oracles G, H', and H that will be simulated by our reduction $\mathcal{R}$. The algorithm $\mathcal{R}$ keeps records a list of query/answer pairs for each of the oracle, i.e., by $Q_G, Q_{H'}$, and Q_H we denote the initially empty lists of query/answer pairs to the oracle G, H', and H, respectively. We assume, without loss of generality, that whenever $\mathcal{A}$ queries H' on $(x\|m)$, it has previously queried G on x. Now, we categorize different types of queries that $\mathcal{A}$ makes to the decryption oracle. Note, that any invocation of the decryption oracle defines the values (s, t, x) and m by simply following the decryption algorithm.

Invalid A decryption query is *invalid* if the output of the decryption oracle is $\perp$.

Valid A decryption query that is not invalid is *valid.*

Likely to be Invalid If $\mathcal{A}$ has not queries $H'(x\|m)$ or $H(s)$ before invoking the decryption oracle, then the query is *likely to be invalid.*

Exceptional If a query that is likely to be invalid turns out to be valid, then we call this query *exccptional.*

The first step of the proof is to show that $\mathcal{A}$ even an unbounded adversary can ask exceptional queries only with negligible probability.

Claim 2 *Let $\mathcal{A}$ be an unbounded adversary that asks only a polynomial number of queries to its oracles. The probability that $\mathcal{A}$ asks an exceptional query is negligible.*

The intuition behind this claim is that even an unbounded adversary cannot predict the mapping of the random oracle better than guessing. Since we allow $\mathcal{A}$ to be unbounded, we can simply assume that whenever $\mathcal{A}_1$ gets the (challenge) ciphertext, it can easily recover s and t by computing $f^{-1}(y)$. We fix the following notation: y^* is the challenge ciphertext that defines the value $s^*\|t^* = f^{-1}(y)$, s_1^*, s_2^*, x^*, t^* and the corresponding message m^*. Obviously, the message m^* is hidden in an information theoretic sense as long as $\mathcal{A}$ never queries G on x^*.

Now, we focus on an arbitrary decryption query y that $\mathcal{A}$ makes *after* seeing the challenge ciphertext y^* and we assume that this query is likely to be invalid. As before, we denote by $s\|t = f^{-1}(y), s_1, s_2, x, t$, and m the corresponding values. Since this query is likely to be invalid, the adversary $\mathcal{A}$ has either not queried H' on $(x\|m)$ or H on s. The following case-by-case analysis shows that y is invalid with all but negligible probability:

Case 1: $\mathcal{A}$ has not queried $H'(x\|m)$ and $x = x^*$ and $m = m^*$: Observe that $s_1 = s_1^*$, because $(x, m) = (x^*, m)$. Whenever the ciphertext is valid, then $s_2 = s_2^*$ and hence $t = t^*$ as well. This, however, means that $y = y^*$, which makes the query not legitimate.

Case 2: $\mathcal{A}$ has not queried $H'(x\|m)$ and $x \neq x^*$: Since $x \neq x^*$ and because $\mathcal{A}$ has not queried $H'(x\|m)$, the value $H'(x\|m)$ is uniformly distributed. Thus, the probability that $\mathcal{A}$ predict this value is, which is necessary in order to pass the check, is $2^{-\|s_2\|} = 2^{-k_1}$, which is negligible.

Case 3: $\mathcal{A}$ has not queried $H'(x\|m)$ and $m \neq m^*$: Analogously to the second case.

Case 4: $\mathcal{A}$ has not queried $H(s)$ and $s = s^*$: If the query is legitimate, then it must hold that $y \neq y^*$, and thus, $t \neq t^*$ as well, which means that $x \neq x'$. However, in order to pass the check performed during the decryption it must hold that $H'(x\|m) = s_2 = s_2^* = H(x^*\|m^*)$. Thus, in order to submit such query $\mathcal{A}$ needs to find a *different* input to the hash function that maps to the same output. Since $\mathcal{A}$ only queries the random oracle a polynomial number of times, it follows easily that this happens only with negligible probability.

Case 5: $\mathcal{A}$ has not queried $H(s)$ and $s \neq s^*$: Since $\mathcal{A}$ never queried H on s, it follows that the value x is uniformly distributed from $\mathcal{A}$'s point of view. Thus, the probability that $\mathcal{A}$ queries $x\|m$ to H' is negligible. Now, this case is identical to the second case.
■

Now, the rest of the proof is similar to the two proofs from the previous lectures and we will only sketch (remind) the main ideas. Recall that the message is information-theoretically hidden if $\mathcal{A}$ does not query G on x^*. Furthermore, the attacker $\mathcal{A}$ must query H on s^* in order to learn x^*. We will now show that the probability that $\mathcal{A}$ queries both H on s^* and G on x^* is negligible (denote this event by hit).

Claim 3 $\text{Prob}[\mathsf{hit}] \approx 0$.

Assume to the contrary that $\text{Prob}[\mathsf{hit}] \not\approx 0$, then there exists an efficient adversary that queries both $H(s^*)$ and $G(x^*)$ with non-negligible probability. Then, we show how to build a algorithm $\mathcal{B}$ with input y^* that inverts the trapdoor permutation also with non-negligible probability. The algorithm $\mathcal{B}$ simulates the random oracle using lazy sampling and it runs a black-box simulation of $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$. At some point, $\mathcal{A}_0$ outputs two messages (m_0, m_1). The algorithm $\mathcal{B}$ then runs $\mathcal{A}_1$ on y^* and simulates decryption queries as follows: Whenever $\mathcal{B}$ receives a query y, then is searches through all queries to H' (that are stored in $Q_{H'}$) and computes for each query (x_i, m_i) the value

$$s_i \leftarrow (G(x_i) \oplus m_i)\|H'(x_i\|m_i).$$

Now, if $s_i \notin Q_H$, then $\mathcal{A}$ has never queried H on (s_i) and $\mathcal{B}$ returns $\perp$. Otherwise, the algorithm $\mathcal{B}$ computes $t_i \leftarrow H(s_i) \oplus x_i$ and checks if $y = f(s_i\|t_i)$. If this is the case, then $\mathcal{B}$ returns $s_i\|t_i$ as the decryption. Otherwise, $\mathcal{B}$ outputs $\perp$.

At some point, $\mathcal{A}_1$ either outputs a bit b or aborts. In any case, the algorithm $\mathcal{B}$ goes through the lists Q_H and Q_G and checks if the pre-image of y has been queried to the oracle. That is, for each $s_i \in Q_H$ and each x_j in Q_G it computes $t_{i,j} \leftarrow x_j \oplus H(s_i)$ and checks if $f(s_i\|t_{i,j}) = y^*$. If there exists a pair that passes this check, then $\mathcal{B}$ outputs the corresponding pair. Otherwise, it aborts.

Now, the proof follows from the following facts: The algorithm $\mathcal{A}$ only notices the difference between this simulation and the one in the real world if it makes an exceptional query to the decryption oracle. However, we have seen that this probability is negligible. Thus, the probability that hit happens in this game is the same as in a real execution. But then, the algorithm $\mathcal{B}$ inverts the TDP $\mathcal{A}$ with the same probability. Since he have initially assumed that the TDP is one-way, the opposite must be true. $\blacksquare$

References

[1] M. Bellare and P. Rogaway, Optimal Asymmetric Encryption, EUROCRYPT'94.

[2] Dan Boneh, Simplified OAEP for the RSA and Rabin functions, CRYPTO'01.

[3] E. Fujisaki, T. Okamoto, D. Pointcheval, J. Stern, RSA-OAEP Is Secure under the RSA Assumption. CRYPTO'01.

[4] Victor Shoup, OAEP Reconsidered, CRYPTO'01.

[5] J. Katz, Lecture Notes *Advanced Cryptography* CMSC 858K, Spring 2004.