CMSC 858K — Advanced Topics in Cryptography March 16, 2004

Lecture 15

Lecturer: Jonathan Katz *Scribe(s): Avi Dalal, Abheek Anand, Gelareh Taban*

1 Introduction

In the previous lecture, we introduced the notion of message authentication: Given message $m \in \mathbb{F}_q$, to authenticate it pick two random secrets $a, b \in \mathbb{F}_q$ and output $(m, am + b)$. The possibility of an attacker outputting (m', t') such that $(m' \neq m)$ and $(t = am' + b)$ is at most $1/q$. The security of this message authentication protocol is information-theoretic and does not rely on any computational assumptions (a proof was given last time). For future reference, we let $\mathsf{Mac}_{a,b}(m) \stackrel{\text{def}}{=} am + b$.

We will use this message authentication scheme to modify the encryption scheme given previously and make it secure against adaptive chosen-ciphertext attacks in the random oracle model. We also introduce OAEP$^+$ and prove its security.

2 The Modified Encryption Scheme

For simplicity, we assume that messages to be encrypted lie in some field $\mathbb{F}_q$ with $|q| = k$ (i.e., the security parameter), and also assume that H maps elements in the domain of the trapdoor permutation family to elements in $\mathbb{F}_q^3$. (If you like, you can think of messages as strings of length ℓ and set $q = 2^\ell$.)

$\underline{\mathsf{Gen}(1^k)}$	$\underline{\mathcal{E}_{pk}(m)}$	$\underline{\mathcal{D}_{sk}(\langle y, C, t\rangle)}$
Generate f, f^{-1}	$r \leftarrow \{0,1\}^k$	$r = f^{-1}(y)$
$pk = f, sk = f^{-1}$	let $H(r) = (a, b, c) \in \mathbb{F}_q^3$	$(a, b, c) = H(r)$
output pk, sk	$C = m + c$	if $aC + b \stackrel{?}{=} t$ then output $C + c$
	$t = \mathsf{Mac}_{a,b}(C)$	else output $\perp$
	output $\langle f(r), C, t\rangle$	

It is not hard to verify that the scheme gives correct decryption.

Theorem 1 *If f is chosen from a trapdoor permutation family, the above scheme is CCA2 secure in the random oracle model.*

Proof We assume the reader is familiar with the proof of semantic security for a related scheme that was given in Lecture 14. The proof here will be similar, but more complicated because we will now need to take into account the *decryption oracle* for an adversary attacking the scheme. Let A be an adversary attacking the scheme, and let r denote the random value used by the sender (i.e., encryption oracle) in constructing the challenge ciphertext $\langle y, C, t\rangle$ that is given to A. Let query be the event that A make the query

$H(r)$ at some point during the experiment, and let dec be the event that A submits a ciphertext $\langle y, C', t'\rangle$ with $(C', t') \neq (C, t)$ but where this ciphertext is decrypted properly (i.e., decryption does not result in $\perp$).

Since we are in the random oracle model, A can only gain any information about the encrypted message if either query or dec occur; thus, as in the proof given previously:

$$\mathsf{Adv}_A(k) \leq \frac{1}{2}\Pr[\mathsf{query} \vee \mathsf{dec}].$$

Define $H(r) = (a^*, b^*, c^*)$. Now, if query has not yet occurred then the only information A has about (a^*, b^*) is that $\mathsf{Mac}_{a^*,b^*}(C) = t$. But then the properties of the message authentication code imply that the probability that dec occurs in any particular query to the decryption oracle is at most $1/q$ (note that every "message" has a unique tag, so setting $C' = C$ will not help). Let query^{1st} denote the event that query occurs before dec (including the case when dec does not occur at all) and define dec^{1st} similarly. The above shows that if A makes at most q_d queries to the decryption oracle we have $\Pr[\mathsf{dec}^{1st}] \leq q_d/q$. Putting everything together we see:

$$\begin{aligned}
\mathsf{Adv}_A(k) &\leq \frac{1}{2}\Pr[\mathsf{query} \vee \mathsf{dec}] \\
&= \frac{1}{2} \cdot \left(\Pr[\mathsf{query}^{1st}] + \Pr[\mathsf{dec}^{1st}]\right) \\
&\leq \frac{1}{2} \cdot (\Pr[\mathsf{query}^{1st}] + q_d/q).
\end{aligned}$$

For A a PPT algorithm, q_d is polynomial and thus q_d/q is negligible (this is why we required $|q| = k$). To complete the proof, we show that $\Pr[\mathsf{query}^{1st}]$ is negligible.

Let A be a PPT adversary attacking the scheme who is given access both to the random oracle $H(\cdot)$ as well as a decryption oracle $D_{sk}(\cdot)$. We construct the following adversary B who will try to invert f on a given point chosen at random from the domain of f. As in the previous proof, B will simulate the experiment for A but this now includes simulating A's access to the decryption oracle (since B does not know $sk = f^{-1}$ we represent the decryption oracle by D). The oracle queries of A are answered in such a way as to ensure consistency between the answers given by the different oracles. This is done by storing two lists: list S_H contains tuples (r, a, b, c) such that $H(r) = (a, b, c)$ (as chosen by B), while list S_y contains tuples (y, a, b, c) such that $H(f^{-1}(y)) = (a, b, c)$ but the important point is that B may not know $f^{-1}(y)$. We now provide a complete description:

$\underline{B(f, y)}$
$S_H = \emptyset; S_y = \emptyset$
run $A^{D(\cdot),H(\cdot)}(f)$ until it outputs m_0, m_1
  (answering queries to D and H as discussed below)
$C, t \leftarrow \mathbb{F}_q$
run $A^{D(\cdot),H(\cdot)}(f, \langle y, C, t\rangle)$ until it halts
  (answering queries to D and H as discussed below)

To answer query $H(r_i)$:

if $f(r_i) = y$ output r_i and halt the experiment
if $r_i = r_j$ for some $(r_j, a_j, b_j, c_j) \in S_H$ then return (a_j, b_j, c_j)
if $f(r_i) = y_j$ for some $(y_j, a_j, b_j, c_j) \in S_y$ then return (a_j, b_j, c_j)
otherwise, choose $(a, b, c) \leftarrow \mathbb{F}_q^3$ and return (a, b, c)
store r_i and the returned values in S_H

To answer query $D(\langle y_i, C_i, t_i \rangle)$:

if $y_i = y$ return $\perp$
if $y_i = y_j$ for some $(y_j, a_j, b_j, c_j) \in S_y$ then decrypt using (a_j, b_j, c_j)
if $f(r_j) = y_i$ for some $(r_j, a_j, b_j, c_j) \in S_H$ then decrypt using (a_j, b_j, c_j)
otherwise, choose $(a, b, c) \leftarrow \mathbb{F}_q^3$ and decrypt using (a, b, c)
store y_i and the (a, b, c) values used in S_y

(Note: "decrypt $\langle y, C, t \rangle$ using (a, b, c)" simply means to return $C + c$ if $aC + b \stackrel{?}{=} t$, and $\perp$ otherwise.) Clearly, B runs in polynomial time when A does; also, it is easy to see that B succeeds in inverting f whenever query occurs and, in particular, if query^{1st} occurs. The above simulation is perfect unless event dec or query occurs. Since we are interested in the event query^{1st} — which occurs immediately if query occurs first and can no longer occur if dec occurs first — the probability of event query^{1st} is the same in the above experiment as in a real execution of A when attacking the encryption scheme. Thus, the security of the trapdoor permutation family implies that $\Pr[\mathsf{query}^{1st}]$ is negligible, as desired. ■

3 Optimal Asymmetric Encryption Padding (OAEP) and OAEP$^+$

A possible drawback of the above scheme is its ciphertext length. Given a trapdoor permutation f acting on k-bit strings, it would be nice to be able to send a ciphertext which is exactly k bits long. OAEP was designed to do this while allowing the message to be as long as possible (and while still being secure against chosen-ciphertext attacks).

OAEP was proposed by Bellare and Rogaway in 1994 [1] and is defined for any trapdoor permutation family. However, the proof was later found to have a subtle error and a number of fixes were proposed (see [4] for a good discussion of the flaw, and a counterexample which illustrates that the flaw is real). Fujisaki, et al. [3] and Shoup [4] show that OAEP is in fact secure when RSA is used as the underlying trapdoor permutation family; the proof of security relies on specific algebraic properties of RSA and does not hold for an arbitrary trapdoor permutation. Boneh [2] gave a simplified version of OAEP which is provably-secure when the RSA or Rabin trapdoor permutation families are used. Shoup [4] showed a way to modify OAEP so as to be secure for an arbitrary trapdoor permutation family. We will present this last scheme (called OAEP$^+$) here both because of its generality and also because it has what is (arguably) the simplest proof.

Let f be a one-way trapdoor permutation, acting on k-bit strings. Also let k_0, k_1 be two parameters such that $k_0 + k_1 < k$ and 2^{-k_0} and 2^{-k_1} are negligible. For example, in an asymptotic setting one could take $k_0 = k_1 = k/3$; more concretely, if RSA is used and

$k = 1024$, then we may set $k_0 = k_1 = 128$. The scheme encrypts messages $m \in \{0,1\}^n$ where $n = k - k_0 - k_1$. The scheme also makes use of three functions:

$$\begin{aligned} G &: \{0,1\}^{k_0} \to \{0,1\}^n \\ H' &: \{0,1\}^{n+k_0} \to \{0,1\}^{k_1} \\ H &: \{0,1\}^{n+k_1} \to \{0,1\}^{k_0}. \end{aligned}$$

These three functions will be modeled as independent random oracles in the security analysis. The scheme is defined as follows:

$\underline{\mathsf{Gen}(1^k)}$	$\underline{\mathcal{E}_{pk}(m)}$	$\underline{\mathcal{D}_{sk}(y)}$
Generate f, f^{-1}	$r \leftarrow \{0,1\}^{k_0}$	$s\|\|t = f^{-1}(y)$,
$pk = f, sk = f^{-1}$	$s = (G(r) \oplus m) \,\|\|\, H'(r\|\|m)$	(where $\|s\| = n + k_1$)
output pk, sk	$t = H(s) \oplus r$	$r = H(s) \oplus t$
	$y = f(s\|\|t)$	parse s as $s_1\|\|s_2$,
	output y	(where $\|s_1\| = n$; $\|s_2\| = k_1$)
		$m = G(r) \oplus s_1$
		if $(H'(r\|\|m) \stackrel{?}{=} s_2)$ output m
		else output $\perp$

The intuition is that this scheme is constructed such that an eventual *simulator*, who does not know sk, is able to answer the decryption queries of an adversary A based only on the oracle queries made by A.

Theorem 2 *If f is chosen from a trapdoor permutation family, the above scheme is CCA2 secure in the random oracle model.*

Proof The proof given here is organized a little differently from the proof given in [4], and the reader is advised to look there for much more detail. Let A be an adversary attacking the scheme. As usual, A will have access to the random oracles in addition to the decryption oracle (and the encryption oracle as well). We assume without loss of generality that whenever A makes a query $H'(r||m)$ it has previously made the query $G(r)$. Let S_G, S_H, and $S_{H'}$ be the set of points at which A has queried G, H, and H', respectively. (These sets grow dynamically each time A queries one of its oracles.) We begin by proving a claim regarding the decryption queries made by A. If a decryption query made by A results in response $\perp$, we say the query is *invalid*; queries which are not invalid are called *valid*. Note that any decryption query y made by A (implicitly) defines values s, t, r, and m (just by following the decryption process); we say a decryption query y is *likely to be invalid* if, at the time the query was made, either A had not yet queried $H'(r||m)$ or A had not yet queried $H(s)$ (for the r, m, s associated with y). Finally, we say a query is *exceptional* if it is likely to be invalid but is, in fact, valid. Then:

Claim 3 *Even if A is all-powerful (but can only make polynomially-many queries to its oracles), the probability that A makes an exceptional query is negligible.*

Proof (of Claim 3): Note that this is an information-theoretic argument based on A's lack of knowledge about the values of the random oracle on points it has not (yet) queried. Since A is all-powerful, we may as well dispense with f, f^{-1} and simply assume that when

A gets the challenge ciphertext y^* it immediately recovers $s^*||t^* = f^{-1}(y^*)$ and that when A submits a decryption query y it already knows $s||t = f^{-1}(y)$.[1] Let m^* be the message encrypted to give the challenge ciphertext (note that even an all-powerful A does not know m^* unless it queries $G(r^*)$), and let s_1^*, s_2^*, r^*, t^* be defined in the natural way based on y^*. We focus on a particular decryption query y that A makes *after* getting the challenge ciphertext (with s_1, s_2, r, t defined in the natural way), and show that the probability that y is exceptional is negligible. Since A makes at most polynomially-many decryption queries, this suffices to prove the claim.

Consider the query y where $s||t = f^{-1}(y)$, and assume that y is likely to be invalid (recall, this is either because A has not queried $H'(r||m)$ or because A has not queried $H(s)$). We show that y is invalid with all but negligible probability by considering the possible cases:

Case 1: A has not queried $H'(r||m)$ and $r = r^*$ and $m = m^*$. Since $(r, m) = (r^*, m^*)$, we also have $s_1 = s_1^*$. If the ciphertext is not invalid, then we must have $s_2 = s_2^*$ and hence $t = t^*$ as well. But this would imply that $y = y^*$, and A is prohibited from querying the decryption oracle with the challenge ciphertext.

Case 2: A has not queried $H'(r||m)$ and $r \neq r^*$. In this case, the value of $H'(r||m)$ is completely random given A's view of the experiment (note that $H'(r||m)$ was not queried during the course of constructing the challenge ciphertext, either). Thus, the probability that y is valid is the probability that $H'(r||m)$ is equal to s_2, which is $2^{-|s_2|} = 2^{-k_1}$ and hence negligible.

Case 3: A has not queried $H'(r||m)$ and $m \neq m^*$. The argument in this case is exactly as in the previous case, so we omit it.

Case 4: A has not queried $H(s)$ and $s = s^*$. Since we must have $y \neq y^*$, this implies that $t \neq t^*$ and hence $r \neq r^*$. The only way y can be valid is if $H'(r||m) = s_2 = s_2^*$, where $s_2^* = H'(r^*||m^*)$. Thus, y is valid only if A has managed to find a *different* input hashing to the same k_1-bit value s_2^*. Since A makes only polynomially-many queries to H', this occurs with only negligible probability.

Case 5: A has not queried $H(s)$ and $s \neq s^*$. In this case, the value of $H(s)$ is completely random from the point of view of A (note that $H(s)$ was not queried when the challenge ciphertext was constructed, either). Thus, the value of r is completely random from the point of view of A, and so the probability that A has queried $H'(r||m)$ is negligible. Assuming A has not queried $H'(r||m)$, we reduce to one of the cases considered previously.
$\square$

Given the above claim, we now prove the theorem in a manner similar to the proof of Theorem 1 (as well as the proof given in the previous lecture). We will be a little informal from now on, but the reader should be able to fill in the missing details (indeed, the difficult part of the proof is the above claim). Note that A has no information about the message that was encrypted to give the challenge ciphertext unless it queries $G(r^*)$. Also, the probability

[1] One may wonder why A needs to submit a decryption query if it is all powerful. The point is that in this claim we are interested in the probability a particular event which is independent of the security of the encryption scheme (indeed, if A is all-powerful than it can "break" the encryption scheme anyway). This claim will be used below to prove the actual security of the scheme for a PPT A.

that A queries $G(r^*)$ without first querying $H(s^*)$ is negligible (since A has no information about r^* until it queries $H(s^*)$, and A makes only polynomially-many queries to G). The preceding two statements are true even if A is all-powerful. So, letting query be the event that A queries both $H(s^*)$ and $G(r^*)$ we have:

$$\mathsf{Adv}_A(k) \leq \Pr[\mathsf{query}] + negl(k).$$

We show that $\Pr[\mathsf{query}]$ is negligible by giving an informal description of a PPT algorithm B which uses A as a subroutine and tries to invert f on a given point y^* chosen at random from the domain of f. B will simulate the random oracle queries of A in the natural way, and when A submits messages (m_0, m_1) to its encryption oracle, B returns the challenge ciphertext y^* to A. More interesting is B's simulation of the decryption oracle for A (recall that B does not know how to compute f^{-1}): upon receiving decryption query y, B searches through the list $S_{H'}$ of queries that A has made thus far to H'. For each $(r_i, m_i) \in S_{H'}$, B first computes

$$s_i = (G(r_i) \oplus m_i)||H'(r_i||m_i).$$

Next, if $s_i \notin S_H$ (i.e., A has not queried $H(s_i)$), B returns $\perp$. Otherwise, B computes $t_i = H(s_i) \oplus r_i$ and then checks whether $y \stackrel{?}{=} f(s_i||t_i)$ (note that B *can* evaluate f in the forward direction). If this test succeeds for a particular pair (r_i, m_i), then B returns m_i to A as the (correct) decryption of y. If the test fails for every i, B returns $\perp$.

At the end of the experiment, B looks through the lists S_H and S_G. For each $s_i \in S_H$ and $r_j \in S_G$, B computes $t_{i,j} = r_j \oplus H(s_i)$ and checks whether $f(s_i||t_{i,j}) \stackrel{?}{=} y^*$. If this is true for any pair, then B outputs $s_i||t_{i,j}$ as the (correct) answer.

The proof concludes using the following observations: (1) until query occurs, the only difference between the view of A in a real experiment and the view of A as simulated by B occurs when A makes an exceptional query (since, in this case, B returns $\perp$ but the decryption query was valid). However, by the claim proven earlier, this occurs with only negligible probability. Thus, (2) the probability of query in the experiment as simulated by B is negligibly close to $\Pr[\mathsf{query}]$ (i.e., the probability of query in the real experiment). Finally, (3) B succeeds in inverting y^* whenever query occurs. Since f is assumed to be a trapdoor permutation family, putting the above observations together shows that $\Pr[\mathsf{query}]$ is negligible. ■

References

[1] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption — How to Encrypt with RSA. Eurocrypt '94.

[2] D. Boneh. Simplified OAEP for the RSA and Rabin Functions. Crypto 2001.

[3] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is Secure Under the RSA Assumption. Crypto 2001.

[4] V. Shoup. OAEP Reconsidered. Crypto 2001.