Public-key Encryption January 23, 2013

Lecture 10

Lecturer: Dominique Schröder

1 The Random Oracle Model

In the previous lectures, we have seen two approaches to build a CCA2 secure encryption scheme. The first one was the DDN construction, which generically combines a CPA secure encryption scheme with a digital (one-time) signature scheme and an adaptively secure NIZK. The second approach was the Cramer-Shoup encryption scheme that is secure under the DDH assumption. Thus, if we want CCA2 security, then we either have to use a rather inefficient scheme or a construction that is based on specific number theoretic assumptions that might not hold in reality. In both cases, however, we know that even the Cramer-Shoup encryption scheme is not as efficient as many CPA secure schemes (e.g., the El Gamal encryption scheme).

Another approach to obtain a more efficient instantiation is to introduce a new cryptographic *model*. We stress that introducing a new model *is not the same* as introducing a new assumption. In this lecture, we discuss a quite popular model known as *the random oracle* (RO) model. The RO model has a long history in cryptography and was first formalized by Bellare and Rogaway [1]. The RO model assumes the existence of a public oracle H that implements a truly random function.

It is not hard to see that such an oracle cannot exist in reality, but this model provides a formal methodology to design and validate the security of cryptographic schemes following a typical two step approach [2]:

1. The first step is the design of the scheme and providing a proof of security in the RO model. The construction might be based on "standard" cryptographic assumptions.

2. To use the scheme in the real world, each party uses a *real-world* hash function (such as e.g., SHA1) and we adjust the scheme appropriately. That is, whenever the scheme asks to evaluate the RO on a value x, then the function is computed locally.

The hope is that the hash functions emulates the RO well enough such that the security proof carries over to the standard model. Currently, there is no theoretical foundation that justifies this view. In fact, there exist (contrived) schemes that are provably secure in the RO model, but completely insecure in the standard model *no matter how the random oracle is instantiated* [2, 3]. From a practical point of view, however, is not clear what it means to emulate a RO well enough and there is not

good understanding if this is an achievable goal. Instead, one can see a proof in the RO model as providing (strong?) evidence that a scheme has no "inherent design flaws". In particular, a proof in the RO model does *not mean* that the scheme is provably secure in the real world.

1.1 Defining the RO Model

The random oracle H can be seen as black box that is accessible by everyone, the honest and the malicious parties. Whenever it is queried with a binary string x, then it outputs binary string $y = H(x)$. The box answers consistently, i.e., whenever some party queries x, it outputs the same response y. The queries to the box are hidden from the other parties, which means that if the party P queries x to H, then the party P' neither learns the query nor its answer. This models the fact that a hash function should be a publicly known function that is computable by everyone. Since internal behavior of the box is unknown and inscrutable, no party knows the complete function (in fact, each party only knows the mapping of the points it has queried).

In what follows, we describe two *equivalent* ways to model the RO model. The first possibility is to choose a function H uniformly at random from the set of all appropriate functions (i.e., those that have the same input and output length). The second way is to *program* the function "on-the-fly".

Random Choice of H: Consider the set of functions that map λ-bit input to $\ell(\lambda)$-bit output. Each of these functions can be represented by a table where for each possible value $x \in \{0,1\}^\lambda$ the corresponding output value is $H(x) \in \{0,1\}^{\ell(n)}$. If we order the inputs lexicographically, then any of these functions can be represented by a string of length $\ell(\lambda) \cdot 2^\lambda$. We can count all possible functions mapping λ-bit inputs to $\ell(\lambda)$-bit outputs, obtaining a total number of $U := 2^{\ell(\lambda)\cdot 2^\lambda}$ different functions. Thus, choosing a random function means to pick a function of U uniformly at random.

Programming H: Alternatively, one can define the random oracle "on-the-fly". That is, we imagine the function H to be defined by a table (as before) mapping λ-bit inputs to $\ell(\lambda)$-bit outputs, but now this table is initially empty. Thus, whenever a party queries H on a value x, then the oracle first checks if x is already in the table. If this is the case, then it simply returns the corresponding entry $H(x) = y$. Otherwise, if x has never been queried before, then the oracle picks a value y uniformly at random from $\{0,1\}^{\ell(\lambda)}$, it stores the mapping (x, y) in the table, and returns $H(x) := y$.

1.2 Security Proofs in the RO Model

In the previous lectures we have seen several definitions and proofs in the standard model. The typical structure of a construction and security proof in the standard

model consist of the definition of a schemes Π and a security proof w.r.t. some experiment $\mathbf{Exp}^{\Pi}_{\mathcal{A}}$. The scheme Π is secure, if the probability that the experiment $\mathbf{Exp}^{\Pi}_{\mathcal{A}}$ outputs some value α is only negligibly bigger than γ, where γ is the maximum desired probability that some "bad" event happens. In the case of CPA/CCA security $\gamma = 1/2$. Here, the probabilities are taken over the random choice of Π and those of $\mathcal{A}$. The parties that use the scheme Π in the real-world will make random choices which guarantees the security of the scheme in the real-world.

Now, if we consider definitions and proof in the RO model, then the situation is different because we have to compute the success probability also *over the random choice of H*. More precisely, in the RO model, the scheme (and also the attacker $\mathcal{A}$) gets black-box access to random oracle H. Let's denote the corresponding construction by Π^H (resp. $\mathcal{A}^H$). Then, the security definition says that a scheme Π^H is secure, if the probability that $\mathbf{Exp}^{\Pi^H}_{\mathcal{A}^H}$ outputs α is only negligibly bigger than γ. The difference is that the probability in this case is *also taken over the random choice of H*. In particular, this means that once the hash function is fixed, the above security claim *does not hold anymore*. This is analogous to the fact that the security claims do *not* hold for a particular set of random choices made by the honest parties, but only with high probability over these choices [2].

Another issue why the security proofs might no longer hold in the standard model is that once the hash functions is fixed, the adversary does not need to query it in a black-box way, but it might look at the code.

2 Public Key Encryption in the RO Model

2.1 Semantically-Secure Encryption

We illustrate the power of the RO model by revisiting the semantically secure scheme based on any trapdoor permutation. Recall that given a trapdoor permutation $\mathcal{F} = \mathsf{Gen}_{td}$, the public-key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ was defined as follows:

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

$\underline{\mathsf{Enc}(ek, m)}$
parse $m \in \{0,1\}$
$x \leftarrow \{0,1\}^\lambda$
$y \leftarrow f(x)$
$b \leftarrow \mathsf{hc}(x) \oplus m$
return $c = (y, b)$

$\underline{\mathsf{Dec}(dk, c)}$
parse c as (y, b)
$x \leftarrow f^{-1}(y)$
$m \leftarrow \mathsf{hc}(x) \oplus b$
output m

where hc is the hardcore predicate for $\mathcal{F}$. Thus, one evaluation of the trapdoor permutation is used to encrypt a single bit. We have learned from the midterm that

encrypting more bits with a single evaluation of the TDP is quite difficult.

In the random oracle, however, we can exploit the fact that the output $H(x)$ is *truly random* if H is a random oracle and the adversary has never queried $H(x)$. This allows us to extract an unbounded number of hardcore bits from H. Obviously, this cannot be true in the standard model: From an information theoretic point of view, once H is fixed $\mathsf{hc}(x)$ completely determines x and thus $H(x)$. Therefore, the entropy of $H(x)$, given $\mathsf{hc}(x)$ is 0. Also note that if the attacker is given $\mathsf{hc}(x)$ then the probability that he queries x to H is negligible, or the attacker inverts the hc.

Given a trapdoor permutation $\mathcal{F} = \mathsf{Gen}_{td}$ and a random oracle H that maps from the domain of the trapdoor permutation family to strings of length ℓ, then the public-key encryption scheme $\mathsf{PKE}^H = (\mathsf{Gen}^H, \mathsf{Enc}^H, \mathsf{Dec}^H)$ is defined as follows:

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

$\underline{\mathsf{Enc}^H(ek, m)}$
parse $m \in \{0,1\}^\ell$
$x \leftarrow \{0,1\}^\lambda$
$y \leftarrow f(x)$
$B \leftarrow H(x) \oplus m$
return $c = (y, B)$

$\underline{\mathsf{Dec}^H(dk, c)}$
parse c as (y, B)
$x \leftarrow f^{-1}(y)$
$m \leftarrow H(x) \oplus B$
output m

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

Proof To prove this theorem, we have to show that the probability that the following experiment evaluates to 1 is negligibly bigger than 1/2,

Experiment $\mathsf{IND\text{-}EAV}^{\mathsf{PKE}}_{\mathcal{A},H}(\lambda)$
$b \leftarrow \{0,1\}$
$x \leftarrow \{0,1\}^\lambda$
$(f, f^{-1}) \leftarrow \mathsf{Gen}_{td}(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}_0^H(f)$
$y_b \leftarrow f(x)$
$B_b \leftarrow H(x) \oplus m_b$
$b' \leftarrow \mathcal{A}_1^H(y_b, B_b)$
Return 1 iff $b' = b$ and $|m_0| = |m_1|$.

In other words, we have to show that

$$\left| \mathrm{Prob}\left[\mathsf{IND\text{-}EAV}^{\mathsf{PKE}}_{\mathcal{A},H}(\lambda) = 1 \right] - \frac{1}{2} \right| \approx 0.$$

The first observation is that if the attacker $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ never queries the random oracle H on the value x, then the output $H(x)$ looks truly random to him. But if

this is the case, then the $H(x) \oplus m$ leaks no information about m and thus, the $\mathcal{A}$ can only guess the bit. In what follows we denote by hit the event that $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ queries H on x in the above experiment and succ is the event that $\mathcal{A}_1$ guesses the bit correctly, i.e., the event that $b = b'$. Using this notations we compute the probability of the equation above as

$$\begin{aligned}
& \left| \mathrm{Prob}\left[\mathsf{IND\text{-}EAV}^{\mathsf{PKE}}_{\mathcal{A},H}(\lambda) = 1\right] - \frac{1}{2} \right| \\
= & \left| \mathrm{Prob}[\mathsf{succ} \mid \mathsf{hit}]\,\mathrm{Prob}[\mathsf{hit}] + \mathrm{Prob}[\mathsf{succ} \mid \neg\mathsf{hit}]\,\mathrm{Prob}[\neg\mathsf{hit}] - \frac{1}{2} \right| \\
= & \left| \mathrm{Prob}[\mathsf{succ} \mid \mathsf{hit}]\,\mathrm{Prob}[\mathsf{hit}] + \frac{1}{2}\,\mathrm{Prob}[\neg\mathsf{hit}] - \frac{1}{2} \right| \\
= & \left| \mathrm{Prob}[\mathsf{succ} \mid \mathsf{hit}]\,\mathrm{Prob}[\mathsf{hit}] - \frac{1}{2}\,\mathrm{Prob}[\mathsf{hit}] \right| \\
\leq & \frac{1}{2}\,\mathrm{Prob}[\mathsf{hit}]\,.
\end{aligned}$$

The last step of the proof is to show that the probability that $\mathcal{A}$ manages to query H on x is negligible.

Claim 2 $\mathrm{Prob}[\mathsf{hit}] \approx 0$.

In the proof of this claim we show that if $\mathrm{Prob}[\mathsf{hit}] \not\approx 0$, then we can build an attacker $\mathcal{B}$ that inverts the one-way property of $\mathcal{F}$ *in the real world* as follows:

$\underline{\mathcal{B}(y, f)}$
$L \leftarrow \emptyset$
$B^* \leftarrow \{0,1\}^\ell$
$(m_0, m_1) \leftarrow \mathcal{A}_0^{\hat{H}}(f)$
$b' \leftarrow \mathcal{A}_1^{\hat{H}}(y^*, B^*)$
Whenever $\mathcal{A}$ queries $\hat{H}$ on some value x, then:
If $f(x) \neq y$:
If $(x, \cdot) \in L$, then return h,
Else return a randomly chosen $h \leftarrow \{0,1\}^\ell$ and store (x, h) in L.
If $f(x) = y$, then output x.

First observe that $\mathcal{B}$ provides a perfect simulation for $\mathcal{A}$ up to the point where $\mathcal{A}$ queries H on x such that $f(x) = y$. This is easy to see because $\mathcal{B}$ simulates the random oracle on all points except for x. Note that $\mathcal{A}_1$ gets a uniformly distributed value B^* as input, but $\mathcal{A}_1$ receives $H(x) \oplus m_b$ in the real game. We argue this does not make any difference from $\mathcal{A}$'s point of view: If $\mathcal{A}_0$ has queried H on x, then $\mathcal{B}$ would have already found the pre-image. Now, assuming that $\mathcal{A}_0$ never asked such a query, means that B^* looks random to $\mathcal{A}_1$ up to the point where $\mathcal{A}_1$ sends x to H. In this case, however, $\mathcal{B}$ learns the pre-image and stops the game. ■

3 CCA2 Secure Encryption

In this section, we discuss two different constructions of a CCA2 secure encryption scheme.

3.1 Preliminaries

We recall the definitions of a message authentication code (MAC).

Definition 1 A message authentication code is a triple of PPT algorithms $\mathsf{MAC} = (\mathsf{MGen}, \mathsf{Mac}, \mathsf{Vrfy})$, where

$\mathsf{MGen}(1^\lambda)$**:** The key generation algorithm takes as input the security parameter 1^λ and returns a key k.

$\mathsf{Mac}(k, m)$**:** The tag generation algorithm takes as input a key k and a message $m \in \{0,1\}^*$, and outputs a tag t.

$\mathsf{Vrfy}(k, m, t)$**:** The deterministic verification algorithm takes as input a key k, a message m, and a tag t. It outputs a bit b, with $b = 1$ meaning valid and $b = 0$ meaning invalid.

A message authentication code is *complete* if for all λ, all $k \leftarrow \mathsf{MGen}(1^\lambda)$, and all $m \in \{0,1\}^*$ we have $\mathsf{Vrfy}(k, m, \mathsf{Mac}(k, m)) = 1$. ♢

Definition 2 A message authentication code $\mathsf{MAC} = (\mathsf{MGen}, \mathsf{Mac}, \mathsf{Vrfy})$ is a existentially unforgeable one-time message authentication code if for all (possibly stateful) PPT algorithms $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ the probability that the experiment $\mathsf{EU\text{-}CMA}^{\mathsf{MAC}}_{\mathcal{A}}$ evaluates to 1 is negligible (in the security parameter λ), where

Experiment $\mathsf{EU\text{-}CMA}^{\mathsf{MAC}}_{\mathcal{A}}(\lambda)$
$k \leftarrow \mathsf{MGen}(1^\lambda)$
$m \leftarrow \mathcal{A}_0(1^\lambda)$
$t \leftarrow \mathsf{Mac}(k, m)$
$(m', t') \leftarrow \mathcal{A}_1(t)$
Return 1 iff $\mathsf{Vrfy}(k, m', t') = 1$ and $m' \neq m$.

♢

The following construction is an *information theoretically* secure one-time MAC that will be part of our first construction. Let $\mathbb{F}_q$ be the field with q elements. The scheme $\mathsf{MAC} = (\mathsf{MGen}, \mathsf{Mac}, \mathsf{Vrfy})$ is defined as follows:

$\underline{\mathsf{MGen}(1^\lambda)}$	$\underline{\mathsf{Mac}(k,m)}$	$\underline{\mathsf{Vrfy}(k,m,t)}$
$a \leftarrow \mathbb{F}_q$	parse $m \in \mathbb{F}_q$	parse k as (a,b)
$b \leftarrow \mathbb{F}_q$	$t \leftarrow am + b$	return 1 iff $t = am + b$
return $k := (a,b)$	return t	

Concerning security, we prove the following theorem.

Theorem 3 *The scheme* $\mathsf{MAC} = (\mathsf{MGen}, \mathsf{Mac}, \mathsf{Vrfy})$ *as defined above is an information theoretical secure message authentication code.*

Proof Let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ be an unbounded adversary, where $\mathcal{A}_0$ outputs some message $m \in \mathbb{F}_q$ and denote by $t = am + b$ the tag that is giving to $\mathcal{A}_1$. From $\mathcal{A}_1$'s point of view the value a is uniformly distributed in $\mathbb{F}_q$, because for every $a \in \mathbb{F}_q$ there exists a single value $b \in \mathbb{F}_q$ that satisfies the equation $t = am + b$ (the value is $b = t - am$). Thus, any forgery (m', t') of $\mathcal{A}_1$ must satisfy the verification equation, i.e., $m \neq m'$ and $t' = am' + b$. This means that the verification equation holds iff $t' - t = a(m - m')$. Since a is uniformly distributed from $\mathcal{A}_1$'s point of view, the probability that can be bounded as follows:

$$\mathrm{Prob}\left[t' = am' + b\right] = \mathrm{Prob}\left[a = (t' - t)(m - m')^{-1}\right] = \frac{1}{q}.$$

■

References

[1] M. Bellare and P. Rogaway, Random oracles are practical: A paradigm for designing efficient protocols, ACM CCS 1993. Full version available at http://cseweb.ucsd.edu/∼mihir/papers/ro.html.

[2] J. Katz and Y. Lindell, Introduction to Modern Cryptography, Chapman and Hall/CRC Press, 2007.

[3] R. Canetti, O. Goldreich, and S. Halevi, The Random Oracle Methodology, Revisited, ACM Sym. on Theory of Computation (STOC) 1998. Full version available at http://eprint.iacr.org/1998/011.pdf.