Last Revised: 2/13/96.

Copyright ©1996 by Phillip Rogaway.

ECS 227: Modern Cryptography
Lecture 9

Finish Symmetric Encryption; Start Asymmetric Encryption

Phillip Rogaway
8 February 1996

This lecture went slowly through the proof that security in the sense of indistinguishability implies semantic security. Somehow, this took almost all of our time.

7.9 Security in the sense of indistinguishability implies semantic security

We have now described two notions of security. Semantic security directly models strong intuition, but the notion is somewhat complicated to define and work with. Security in the sense of indistinguishability sounds simpler, but the intuition might seem less compelling. In any case, the two notions turn out to be equivalent. We shall prove one of the two directions which demonstrate equivalence.

Theorem 7.9.1 *Suppose adversary* A_* (t,q,μ,ϵ)*-breaks encryption scheme* $\Pi = (\mathcal{E},\mathcal{D},\mathcal{K})$*, in the sense of semantic security, with respect to* $\mathcal{M}$*,* f*, where* $\mathcal{M}$ *is a valid distribution on the message space. Then there exists an adversary* A *that* (t',q',μ',ϵ')*-breaks* Π*, in the sense of indistinguishability, where*

$$\begin{aligned} t' &= t + O(\mu + \max_{x\in\mathcal{M}} |f(x)|) \\ q' &= q+1 \\ \mu' &= \mu + \max_{x\in\mathcal{M}} |f(x)| \\ \epsilon' &= \epsilon \end{aligned}$$

The $O(r)$ notation means $Cr + C'$, for some absolute constants C and C' which depend only on details of the underlying model of computation.

A somewhat more elegant way to state the above is the equivalent:

Theorem 7.9.2 *Suppose encryption scheme Π is (t, q, μ, ϵ)-secure, in the sense of indistinguishability. Then Π is $(t', q', \mu', \epsilon')$-secure, in the sense of semantic security, where*

$$\begin{aligned} t' &= t - O(\mu + \max_{x \in \mathcal{M}} |f(x)|) \\ q' &= q - 1 \\ \mu' &= \mu - \max_{x \in \mathcal{M}} |f(x)| \\ \epsilon' &= \epsilon \end{aligned}$$

Make sure you understand the above way of describing things, and why it's equivalent. Why did the additions become subtractions?

Proof. We have an adversary A_* which $(t,\ q,\ \mu, \epsilon)$-breaks the semantic security of Π with respect to $\mathcal{M}$ and f. Imagine for the moment that it is possible to (1) sample in $\mathcal{M}$ and (2) compute f. We will not really need these abilities; at the end, we will remove this assumption using nonuniformity.

We construct adversary A as follows.

Using the special abilities we just mentioned, adversary A will begin by choosing a random sample x^* from $\mathcal{M}$. Then adversary A will compute $\gamma^* = f(x^*)$.

Next adversary A will apply its oracle, h, to x^*, thereby computing a string $y = h(x^*)$.

Now A simulates the behavior of A_* running on input y. Any time A_* makes an oracle query of some string u we have A make an h-query of the *same* string u, getting some answer w in response. We have A return to A_* this string w. When, finally, A_* would output a string, γ, and halt, adversary A checks if $\gamma = \gamma^*$. If so, A outputs a 1. Otherwise, A outputs a 0. (Recall that an output of 1 amounts to a guess of h being a "good" encryption oracle, while an output of 0 amounts to a guess that h is a "bogus" encryption oracle.)

Ignore the time to compute x^* and γ^*. Then the reader can check that the claimed bounds for t', q' and μ' are easily achieved using the algorithm we've described. Now we must evaluate ϵ'.

First, suppose that A's oracle h is instantiated with an oracle $\mathcal{E}_a(\cdot)$, for a random $a \leftarrow \mathcal{K}$. Then the environment which A provides for A_* is *identical* to the one defining how well A_* finds $f(x)$, so:

$$\Pr\left[a \leftarrow \mathcal{K};\ x \leftarrow \mathcal{M};\ y \leftarrow \mathcal{E}_a(x) :\ A_*^{\mathcal{E}_a(\cdot)}(y) = f(x)\right] \quad = \quad \Pr\left[A^{\mathcal{E}_a(\cdot)} = 1\right] \tag{1}$$

On the other hand, if A's oracle h is instantiated by an oracle $\mathcal{E}_a(\$^{|\cdot|})$, for a random $a \in \mathcal{K}$, then

$$\begin{aligned} \Pr\left[a \leftarrow \mathcal{K};\ x \leftarrow \mathcal{M};\ y \leftarrow \mathcal{E}_a(\$^{|x|}) :\ A_*^{\mathcal{E}_a(\$^{|\cdot|})}(y) = f(x)\right] &= \Pr\left[A^{\mathcal{E}_a(\$^{|\cdot|})} = 1\right] \\ &\leq P^*_{\mathcal{M},f} \end{aligned}$$

because the information A_* receives (both y and the answers from its oracle) is completely uncorrelated with x. Multiplying by -1,

$$-\Pr\left[A^{\mathcal{E}_a(\$^{|\cdot|})} = 1\right] \quad \geq \quad -P^*_{\mathcal{M},f} \ . \tag{2}$$

Now we are given that

$$\Pr\left[a \leftarrow \mathcal{K};\ x \leftarrow \mathcal{M};\ y \leftarrow \mathcal{E}_a(x) :\ A_*^{\mathcal{E}_a(\cdot)}(y) = f(x)\right] \quad \geq \quad \epsilon + P^*_{\mathcal{M},f}$$

so, from, (1),

$$\Pr\left[A^{\mathcal{E}_a(\cdot)} = 1\right] \quad \geq \quad \epsilon + P^*_{\mathcal{M},f} \ .$$

Combining this with (2) we have

$$\Pr\left[A^{\mathcal{E}_a(\cdot)} = 1\right] - \Pr\left[A^{\mathcal{E}_a(\$^{|\cdot|})} = 1\right] \quad \geq \quad \epsilon + P^*_{\mathcal{M},f} - P^*_{\mathcal{M},f} \ ,$$

that is, $\mathrm{Adv}_A \geq \epsilon$.

To remove the technical conditions we assumed about A, realize that Adv_{A_*} is actually the $\mathcal{M}$-weighted sum of the advantages for particular x-values, that is, $\mathrm{Adv}_{A_*} = \sum_{x \leftarrow \mathcal{M}} \Pr_{\mathcal{M}}[\{x\}] \cdot \mathrm{Adv}_{A_*,x}$, where $\mathrm{Adv}_{A_*,x}$ is defined by

$$\mathrm{Adv}_{A_*,x} \quad = \quad \Pr\left[a \leftarrow \mathcal{K};\ y \leftarrow \mathcal{E}_a(x) :\ A_*^{\mathcal{E}_a(\cdot)}(y) = f(x)\right] - P^*_{\mathcal{M},f}.$$

We have simply taken the $x \leftarrow \mathcal{M}$ out of the probablity in the experiment which defines A_*'s advantage. Thus we can choose some *particular* $x^* \in \mathcal{M}$ for which $\mathrm{Adv}_{A_*,x^*} \geq \mathrm{Adv}_{A_*}$, for if the average of a bunch of numbers is λ, one of the numbers in the bunch must be at least λ. Fix such an at-least-the-mean point x^*. Let $\gamma^* = f(x^*)$. Now use the nonuniformity of our adversary A to "hardwire" into her the particular values of x^* and γ^* which she needs to know. Don't have A sample from $\mathcal{M}$ and compute f at the sample; instead, just the particular strings x^* and γ^*. $\diamondsuit$

8 Asymmetric Encryption

8.1 The basic setting

To facilitate the receipt of encrypted messages, party R generates a *public key*, $\mathcal{E}_R$, and a matching *secret key*, $\mathcal{D}_R$. The public key is made publicly known and it is publicly tied, somehow, to R's identity. (Image that the string $(R,\ \mathcal{E}_R)$ is published in the N.Y. Times.) Thus everyone (including the adversary) knows $\mathcal{E}_R$ as well as the binding $R \leftrightarrow \mathcal{E}_R$. When some party, S, wants to send a message x to R, party S computes a point $y \leftarrow \mathcal{E}_R(x)$ and sends y to R. Party

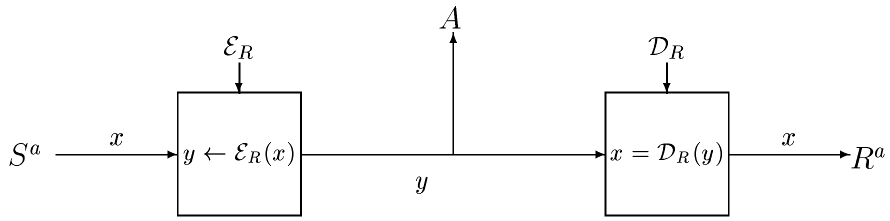

Figure 1: One view of asymmetric encryption. Sender S encrypts messages with the recipient's encryption algorithm $\mathcal{E}_R$. The recipient's matching decryption algorithm, $\mathcal{D}_R$, recovers x.

S recovers x from y by computing $x \leftarrow \mathcal{D}_R(y)$. An adversary A, witnessing y, shouldn't get information about x it is not already in possession of— at least not information beyond the length of x. See Figure 1.

The advantage of asymmetric encryption (also called *public key* encryption) over symmetric encryption is that S and R to not have to have a shared and secret key a. This sounds like an enormous advantage. But S does have to be sure of $R \leftrightarrow \mathcal{E}_R$; obtaining this knowledge becomes the equivalent of coming to possess shared symmetric keys. Asymmetric encryption is also a central building block to obtaining more complex cryptographic aims.

In the previous description we didn't bother to make any distinction between the public key as a string and the algorithm which it specifies; nor did we make any distinction between the secret key as a string and the algorithm which it specifies. You might like to think of these as different. But, in fact, the public key and secret key are understood to be strings which specify an algorithm in a unique way, so the distinction isn't really all that important.

Diffie and Hellman published this notion of a public key encryption scheme (you should have read their paper by now) and they proposed a general idea for achieving it. The suggested that the scheme be specified by a *trapdoor permutation generator*. This is a function $\mathcal{G}$ which, on input 1^k, outputs a pair of permutations, f and f^{-1}, which are inverses of one another.

To directly use a trapdoor permutation generator for public key encryption, a party R chooses an (f, f^{-1}) by running the trapdoor permutation generator, and then R selects f for his public key, keeping f^{-1} as his secret key. To encrypt a message x, a sender S computes and sends to R the string $y = f(x)$. (Note we assume here that the message x is in the domain of f.) Party R recovers x by computing $x = f^{-1}(y)$. This really is x, because f^{-1} is the inverse of f.

Though an encryption scheme and a trapdoor permutation generator might sound similar, they are actually quite different, and you should keep the straight. In fact, there are many ways to achieve an encryption scheme, and the simple manner which we have just described is not, in fact, a very good one. Similarly, there are many uses for trapdoor permutation generators besides asymmetric encryption.

You should be able to spot one reason for why the direct use of a trapdoor permutation generator as an encryption scheme isn't going to be very good: the method is stateless and deterministic. For the exact same reasons that stateless, deterministic encryption didn't achieve our notion of security in the symmetric case, the above suggestion won't achieve a strong notion of security in the asymmetric case either.

8.2 The RSA scheme

When Diffie and Hellman published their idea there was one minor glitch: they hadn't yet been able to come up with any trapdoor permutation generator. They postponed publication as long as they thought advisable, trying to come up with one. But finally they went forward.

Rivest, Shamir and Adleman heard of Diffie and Hellman's work and they set out to find a trapdoor permutation. The finally succeeded. Here is a description, following a bit of review of notation from number theory.

If n is a number then Z_n^* denotes the multiplicative group of integers modulo n. As a set, this is all the numbers between 1 and n which are relatively prime to n, $\mathsf{Z}_n^* = \{1 \leq a < n : \ (a,n) = 1\}$. This is made into a group under the operation of multiplication modulo n. The size of this group, —that is, the number of numbers between 1 and n which are relatively prime to n— is denoted $\varphi(n) = |\mathsf{Z}_n^*|$. (This function φ is called the "Euler totient function.") I leave it for you to check that, when n is the product of distinct primes p and q, $\phi(n) = (p-1)(q-1)$.

You need the following basic facts from number theory. First, "Lagrange's theorem"— for any finite group G, $a^{|G|} = 1$. (We are using multiplicative notion, so 1 is the group's identity element and and a^i means applying the group operation i times.) Second, there is an algorithm —Euclid's algorithm— which can, in time polynomial in $\lg n$, computes from (a,n), where $a \in \mathsf{Z}_n^*$, the inverse of a in Z_n^*.

Now, on to the trapdoor permutation generator known as RSA.

On input k (the security parameter) choose two random but distinct $k/2$-bit prime numbers, p and q. Multiply them to $N = pq$. Also choose a number $e \in \mathsf{Z}_{\varphi(N)}^*$. Find the inverse, d, of e in $\mathsf{Z}_{\varphi(N)}^*$. This is the unique value $d \in \mathsf{Z}_{\varphi(N)}^*$ such that $ed = 1 \bmod \varphi(N)$. This element is found using Euclid's algorithm. The pair of functions (f, f^{-1}) returned by the trapdoor permutation generator is:

$$\begin{aligned} f(x) &= x^e \bmod N, \quad \text{with inverse} \\ f^{-1}(x) &= x^d \bmod N \end{aligned}$$

When used in the most natural way to make a public key encryption algorithm, f is the public key and f^{-1} is the matching secret key.

Let us verify that f and f^{-1}, as described above, really are inverses of one another. We need to check that, for all $x \in \mathsf{Z}_N^*$, $(x^e \bmod N)^d \bmod N = x$. But $(x^e \bmod N)^d \bmod N = x^{ed} \bmod N = x^{k\varphi(N)+1} \bmod N$, for some number k (because d is the inverse of e in $\mathsf{Z}_{\varphi(N)}^*$). But $x^{k\varphi(N)+1} \bmod N = x \cdot (x^{\varphi(N)})^k) \bmod N = x1^k \bmod N = x \bmod N = x$.