Last Revised: 3/5/96.

Copyright ©1996 by Phillip Rogaway.
Original scribe notes prepared by Joel Dodson

ECS 227: Modern Cryptography
Lecture 14

Efficient Asymmetric Encryption (Part II)

Phillip Rogaway
27 February 1996

Today we finished talking about the Optimal Asymmetric Encryption Padding (OAEP) scheme then started the topic of plaintext awareness.

We started with a review of using the Goldreich-Levin hard-core bit to encrypt a string. Namely, for string $m = m_1 m_2 \cdots m_n$, encrypt each bit m_i using the following scheme:

$$\langle f(x_i),\ p_i,\ (p_i \odot x_i) \oplus m_i \rangle\ .$$

where f is a from a trapdoor permutation and x_i and p_i are random strings. This, however, is very inefficient, as was detailed in the lecture 13. The scheme that are actually look like $f(\text{randomstuff} \parallel m)i$, as we described in last lecture.

We then went on to review the OAEP encryption scheme. That's all covered in the lecture 13 notes, so I wont elaborate on that.

9.6 Definition of secure encryption

In this section we'll define a secure asymmetric encryption scheme for the random oracle model. We'll use the find/guess notion of security, but we'll be sensitive to the hash functions G and H which the encryption scheme has access to.

Definition 9.6.1 *Let $\mathcal{G}$ be an encryption scheme and let A be an adversary. Then A is said to (t, q_G, q_H, ϵ)-*break *$\mathcal{G}(1^k)$, if*

$$\begin{aligned} Adv_A(k) \ &\stackrel{\text{def}}{=}\ 2 \cdot \Pr[(\mathcal{E}, \mathcal{D}) \leftarrow \mathcal{G}(1^k);\ (G,H) \leftarrow \Omega; \\ &\qquad (x_0, x_1, c) \leftarrow A^{G,H}(1^k, \mathsf{find}, \mathcal{E});\ b \leftarrow \{0,1\};\ y \leftarrow \mathcal{E}^{G,H}(x_b) : \\ &\qquad A^{G,H}(1^k, \mathsf{guess}, y, c) = b] - 1 \\ &\geq\ \epsilon \end{aligned}$$

and $A(1^k, \cdots)$ makes at most q_G oracle queries to its G-oracle, and and $A(1^k, \cdots)$ makes at most q_H oracle queries to its H-oracle, and the total running time of

$A(1^k, \mathsf{find}, \cdots)$ and $A(1^k, \mathsf{guess}, \cdots)$ is at most t. We demand that, for any (x_0, x_1, c) which $\mathcal{E}(1^k, \mathsf{find}, \cdots)$ might output, $|x_0| = |x_1|$.

In the above definition, c is a state returned by A from its find mode that can be used in its guess stage. Hash functions G and H have been modeled as random functions from the space Ω.

Now recall what it means for an adversary to invert a trapdoor permutation generator $\mathcal{F}$

Definition 9.6.2 *An adversary I is said to (t, ϵ)-*invert *the trapdoor permutation generator $\mathcal{F}(1^k)$ if*

$$\begin{aligned} \mathrm{Adv}_I &\stackrel{\text{def}}{=} \Pr\left[(f, f^{-1}) \leftarrow \mathcal{F}(1^k);\ x \leftarrow \{0,1\}^k;\ y \leftarrow f(x) :\ I(1^k, f, y) = x\right] \\ &\geq \epsilon \end{aligned}$$

and $I(1^k, \cdots)$ runs in at most t time.

Now we can state the theorem that was proven in lecture 13.

Theorem 9.6.1 *Suppose adversary A (t, q_G, q_H, ϵ)-breaks the encryption scheme $\mathcal{G}_{\mathrm{OAEP}}[k, k_0, n, \mathcal{F}]$. Then there is an adversary I that (t', ϵ')-inverts $\mathcal{F}(1^k)$, where*

$$\begin{aligned} t' &= t + q_G \cdot q_H \cdot (\mathit{TIME}_f(k) + O(1)) \\ \epsilon' &= \epsilon \cdot (1 - q_G \cdot 2^{-k_0} - q_H \cdot 2^{-n}) - q_G \cdot 2^{-k+1} \end{aligned}$$

As long as k and k_0 are large enough, the term $q_G \cdot 2^{-k_0}$, $q_H \cdot 2^{-n}$ and $q_G \cdot 2^{-k+1}$ are small enough so that ϵ and ϵ' are very close. On the other hand, t' could be much larger than t due to the $q_G \cdot q_H$ term. It is an open problem to improve the analysis, or to change the scheme in such a way that one can establish a better bound. The scheme of the next section seems likely to have a better bound (though has not yet been shown).

9.7 The Idea of Plaintext Awareness

This section we started toward the end of lecture. Following is just sort of a motivation for the idea of plaintext awareness.

Consider the situation where contractor A sends an encrypted bid stating "I'll do that job for \$100,000." Suppose now that contractor B can get a hold of that ciphertext and change it in such a way that now it says "I'll do that job for \$99,000." Contractor B submits that bid as his own, without actually knowing how much he's bid. Contractor B was able to decrement contractor A's bid, but he hasn't learned anything about contractor A's bid.

If the sort of thing described above happen, we say that the encryption scheme is *malleable.*

Note that the simple deterministic RSA-scheme is highly malleable: Given ciphertext $y_1 = x_1 \bmod N$ and $y_2 = x_2 \bmod N$, an adversary can produce the ciphertext for $x_1 x_2 \bmod N$ by simply computing $y_1 \cdot y_2 \bmod N$. Though this way of encrypting does not satisfy our earlier notions of security, nothing that we have said far implies that a scheme is a secure encryption scheme should be not malleable. In fact, in class we came up with a scheme that was secure in the sense of indistinguishable but which was quite malleable.

What we need is an encryption scheme that is non-malleable.

We will over-achieve. We will describe an encryption scheme which guarantees that an adversary "knows" the plaintext of any ciphertext she produces. For a semantically secure encryption, this implies non-malleability. Informally, if you can produce the ciphertext of a string, x', which is related to the plaintext x of a ciphertext y, and you know x', then you know something of x, too.

We left off with the question of what it means to "know" something? We wanted to define what it means to say that an adversary who produces the ciphertext necessarily "knows" the corresponding plaintext. That's where the next lecture picks up.