Last Revised: 03/02/96.

ECS 227: Modern Cryptography
Lecture 13

Efficient Asymmetric Encryption

Phillip Rogaway
22 February 1996

References for this lecture:

- "PKCS #1: Encryption Standard", RSA Data Security, Inc.
- "Optimal Asymmetric Encryption - How to encrypt with RSA", Bellare and Rogaway 1995.

9 Efficient Asymmetric Encryption

In the last two lectures we proved the correctness of the Goldreich/Levin hard-core bit, and we showed how to use it to provide a provably-secure method for asymmetrically encrypting a message (assuming the existence of a trapdoor permutation). The problem with the resulting encryption scheme is that it is terribly inefficient. For example, in order to encrypt a single bit b using this method, where the underlying trapdoor permutation is $\mathcal{RSA}$ and $N = pq$ is a 1024-bit modulus and the encryption exponent is $e \in \mathsf{Z}^*_{\phi(N)}$, we must first compute $r^e \bmod N$ (a cubic-time operation), then choose a 1024-bit random p, and then the encryption of b is $\mathcal{E}(b) = \langle r^e(N),\ p,\ (p \odot r) \oplus b\rangle$. So $|\mathcal{E}(b)| = 2049$ bits.

When the message x is reasonably long, instead of using an asymmetric scheme on an entire message, we typically encrypt the message under a symmetric encryption scheme (eg., one based on DES) under a new, random symmetric key. Then the symmetric key is encrypted using the asymmetric scheme. The enciphered text is the enciphered symmetric key together with the symmetrically-encrypted x. Using this approach with with RSA and $k = |N| = 1024$ and DES, we would have to send a total of about $2049 \cdot 56 + |x| + 64 = 114808 + |x|$ bits to encrypt x: 2049 bits for each bit of the encrypted DES key, and about $|x| + 64$ for the DES-encrypted message x. This is too many bits, as well as too much time (we had to perform 56 modular exponentiations).

9.1 A First Attempt at Efficient Encryption

Instead of securely encrypting each bit, we could try to use RSA only once. Assume $n = |x| < k$. (This would typically be the case, given what we just said about how asymmetric encryption is usually used.) We consider RSA-encrypting by, say,

$$\mathcal{E}(x) \;=\; (00\,00 \cdots 00\,01 \,\|\, x)^e \bmod N$$

where x is the message, and we simply pad it with zero-bytes (a one-byte to terminate) before exponentiating. The enciphered text is now only $k = |N|$ bits, and only one modular exponent needs to be computed. But does the encryption scheme meet our definition of security?

Most definitely it does not. A glaring problem is that this method is completely deterministic. We've seen that good encryption schemes need to be stateful or probabilistic. Otherwise, much information is leaked by the encryption: for example, an adversary can see if some particular message has been sent twice.

9.2 PKCS#1

In PKCS#1 the above problem is rectified by requiring that messages be formatted as follows. First, we prepend to x: the zero byte 00; followed by the byte 02; followed by at least 8 bytes of random non-zero bytes; followed by the byte 00. We use enough non-zero random bytes to make the entire message come out to k bits. So,

$$\mathcal{E}_{\mathrm{PKCS\#1}}(x) \;=\; (00 \,\|\, 02 \,\|\, \$^t \,\|\, 00 \,\|\, x)^e \bmod N$$

where $t = (k - |x| - 24)/8$, and $\$^t$ denotes a string of t random non-zero bytes.

We see that there are only k bits being sent and that there is again only one modular exponentiation required. In addition, the scheme is not deterministic.

But we desire a scheme which is provably secure under the assumption that the underlying function ($\mathcal{RSA}$, in this case) is a trapdoor permutation. But, there is no reason to believe that the above scheme is secure under the assumption that RSA is a trapdoor permutation. In fact it is easy to construct a "good" trapdoor permutation for which the above encryption scheme (generalized to use an arbitrary trapdoor permutation) *won't* itself be good.

Consider, for example, the possibility that the message x might have a lot of structure. Then it wouldn't be too surprising if $\mathcal{E}_{\mathrm{PKCS\#1}}(x)$ conveys some of that structure. For example, if x_1 is the message consisting of all 1 bits, and x_0 is the message consisting of all 0 bits, $\mathcal{E}_{\mathrm{PKCS\#1}}(x_0)$ might be distinguishable from $\mathcal{E}_{PKCS\#1}(x_1)$: nothing about $\mathcal{RSA}$ being a trapdoor permutation would seem to prevent this possibility.

9.3 The IBM Technique

Some cryptographers at IBM realized that it was a bad idea to allow structure in the message x to show through to the string to which we apply the forward direction, f, of the trapdoor permutation. They decided to try to "mask" the bits of x. First, they would choose a random k_0-bit r. Then they'd apply a pseudorandom generator $G : \{0,1\}^{k_0} \to \{0,1\}^n$ to r. (The generator G might be based on SHA, for example.) Let $s = x \oplus G(r)$. This is the "masked" version of x. Finally, they'd encrypt x as $f(s \parallel r)$. (Eg., in the case of RSA, $(s \parallel r)^e \bmod N$).

To decrypt a ciphertext y, we should first apply f^{-1} to y to get s and r. Then apply G to r to get $G(r)$. Now, $x = s \oplus G(r)$.

Unfortunately, this scheme has the same problem as PKCS#1: it is still possible that we could get some information about x from its encryption. Suppose, for example, that given $f(w)$, it is easy to compute the last k_0 bits of w, and the first $n/2$ bits of w. This is entirely possible: we know, since $\mathcal{F}$ is a trapdoor permutation, that we can't recover *all* of w; but we don't know that we can't easily get a lot of it. Then we'd be able to recover r, compute $G(r)$, and the adversary could compute half the bits of x.

9.4 Optimal Asymmetric Encryption Padding (OAEP)

Bellare and Rogaway finally came up with a provably-secure method as efficient as those above. The provable security is in a somewhat enriched model of computation; we shall see this later on. The idea is to not only to mask the bits of the message x, but also to mask —using the masked bits of x— the the bits of the string r which was used to mask x. This is done as follows. Let x be the message of length n. Pick k_0 bits, r, at random. Apply a pseudorandom generator $G : \{0,1\}^{k_0} \to \{0,1\}^n$ to r. Let $s = x \oplus G(r)$. Apply a hash function $H : \{0,1\}^n \to \{0,1\}^{k_0}$ to s. Then, let $t = r \oplus H(s)$. Finally, the encryption of x is $f(s|t)$. So $\mathcal{E}^{G,H}_{\mathrm{OAEP}}(x) = f(s|t)$ where $s = x \oplus G(r)$ and $t = r \oplus H(s)$.

To decrypt ciphertext y, compute $f^{-1}(y)$ to get s and t. Apply H to s to get $H(s)$. Set $r = H(s) \oplus t$. Apply G to r to get $G(r)$. Finally, compute $x = s \oplus G(r)$.

9.5 Correctness of OAEP

Usually we define our measure of security and then we do our reduction. We've already defined secure encryption (two ways, in fact), and we've already defined trapdoor permutations. But I claim we're not ready for a proof. Why's that?

It's those functions G and H. They're modeling an "idealized hash function" which everyone (including the adversary) has access to. They weren't in our definition of a secure encryption scheme. In this context, they have to be. That is, a formal definition of secure encryption has to take account of the presence of those idealized hash functions.

Let's ignore the above issue for a minute and proceed in the absence of any formal notion of security.

A proof will have to describe how to invert the trapdoor permutation given an adversary that breaks the encryption scheme. We begin with a bit of intuition.

We want to say that if we are missing any part of s or t, then we can't know anything about the message x. (1) Suppose we don't know something about s. Then we don't know anything about $H(s)$. Thus we don't know anything about $r = H(s) \oplus t$. Thus we don't known anything about $G(r)$. Thus we don't know anything about $x = s \oplus G(r)$. (2) Similarly, suppose we don't know something about t. Then we don't know something about $r = t \oplus H(s)$. Then we don't know anything about $G(r)$. Thus we don't know anything about x. So to know something about x, we better know all of s and t. But if we know all of s and t, then we've inverted the trapdoor permutation.

Now, we work this intuition into a proof sketch. We want to show that if there exists an adversary A that breaks OAEP, then there exists an adversary I that can break the trapdoor permutation $\mathcal{F}$. Think back to the definition of an encryption adversary A in which A runs in two stages: a find stage and a guess stage:

1. $A^{G,H}(1^k, \mathsf{find}, \mathcal{E}) \to x_0, x_1, c$. That is, A finds two equal-length messages, x_0 and x_1, which she wants to try to distinguish the encryptions of.

2. $A^{G,H}(1^k, \mathsf{guess}, y, c) \to b'$. The adversary is trying to guess if $y = \mathcal{E}(x_0)$ or $y = \mathcal{E}(x_1)$.

Review this notion if you've forgotten it.

Now we wish to construct a trapdoor inverting adversary I. It gets a description of f (taken from $(f, f^{-1}) \leftarrow \mathcal{F}(1^k)$) and a y (taken according to $y = f(x)$, where $x \leftarrow \{0,1\}^k$). Algorithm I wants to find $x = f^{-1}(y)$.

The natural way to define such an adversary I is like this. Run A. Watch the questions that A asks its oracles (G and H) and piece together the oracle queries to invert f.

In greater detail, here is the behavior of I. Remember that I is given 1^k, f, y.

1. Run $A(1^k, \mathsf{find}, \mathcal{E}_f)$, where $\mathcal{E}_f$ denotes the OAEP encryption function associated to f. Assume without loss of generality that A never asks an oracle query twice. If A asks its G-oracle a question g, make up a random n-bit response, $G(g)$, and return this string to A. Similarly, if A asks its H-oracle a question h, make up a random k_0-bit string $H(h)$ and return that to A. Adversary I will keep track of all the $(g, G(g))$ and $(h, H(h))$ that it invents.

2. Run $A(1^k, \mathsf{guess}, y, c)$. Answer G- and H-oracle calls exactly as above.

3. Finally, A outputs a bit. Ignore that bit. BUT: look to see if there is any (g, h) where g is among the defined $(g, G(g))$, and h is among the defined $(h, H(h))$, for which $f(h \parallel g \oplus H(h)) = y$. If so, output $h \parallel g \oplus H(h)$ — adversary I has inverted f at y.

All we need now is to believe that I is successful as often as M is successful. The proof is contained in the [Bellare, Rogaway] paper. The proof is a slightly technical, because the "world" which I has provided A is not exactly like the defining one. The proof shows that it differs sufficiently little for this to be irrelevant.