8 Encryption: Semantic Security and Practical Issues

8.1 Semantic Security

Recall that for information-theoretic encryption, we had two definitions of security. Shannon secrecy focused on just two messages (much like indistinguishability we defined for public-key encryption), and perfect secrecy focused on obtaining information from encryption of a single messages drawn at random from some distribution. This section defines the analogue of perfect secrecy for public-key encryption.

First of all, because we are interested in computational security, which is usually formulated in terms of asymptotics, we will have multiple distributions on the message space—one for each value of the security parameter k (Shannon didn't have to do this and could consider a single fixed message space, because he had no computational hardness requirements; we can do the same if we formulate everything in terms of concrete security for a particular k, as explained in the lecture on defining next-bit unpredictability for pseudorandom generators). We will restrict messages to be of length polynomial in k, and, because encryption cannot hide length, we will provide the adversary with information on the length of the message chosen. Secondly, we can't require that there should be no information about the plaintext in the ciphertext (of course there will be—in fact, the ciphertext, combined with the public key, uniquely determines the plaintext). Rather, we will say that this information is not usable in polynomial time: whatever function of the plaintext you can compute with the ciphertext you can also compute without it. An finally, we will give the adversary arbitrary auxiliary information it wants about the plaintext (this models information adversary could obtain by other means, such as observing the behavior of various parties, etc.).

More precisely, let S be a randomized function that generates messages given the security parameter k; we require that there exists some polynomial p such that $|S(k)| < p(k)$. Note that we do not require S to be efficiently computable, or even computable at all. This is meant to model the distribution of message that the encryptor wants to send. Let $f, h : \mathbb{N} \times \{0,1\}^* \to \{0,1\}^*$ be functions (not necessarily computable) that take the security parameter k and the message as inputs, and output some string whose length is polynomial in k (i.e., there must exist q such that $|f(k,m)| \le q(k)$ and $|h(k,m)| \le q(k)$). These are meant to model the information that the adversary is interested in, and the information that the adversary already has, respectively. Finally, let A be a probabilistic polynomial-time machine that attempts to compute $f(k,m)$ given $h(k,m)$, the length of m, a public key, and an encryption of m using the public key. We want to say that there is a machine B that computes $f(m)$ without the encryption (thus, only from $h(k,m)$ and the length of m). Consider the following two experiments.

`expA`(k)
1. $m \leftarrow S(k)$
2. $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathrm{Gen}(1^k)$
3. $c \leftarrow \mathrm{Enc}_{\mathrm{PK}}(m)$
4. $x \leftarrow A(1^k, h(k,m), 1^{|m|}, c, \mathrm{PK})$
3. Output 1 if $f(k,m) = x$ and 0 otherwise

`expB`(k)
1. $m \leftarrow S(k)$
2. $x \leftarrow B(1^k, h(k,m), 1^{|m|})$
3. Output 1 if $f(k,m) = x$ and 0 otherwise

Note that B gets no information at all beyond what h and the length of the message reveal (and h, can, in particular, be the constant function that reveals no information). This is exactly the point of secure encryption: without any information you can compute f just as well as with the ciphertext and the public key.

Definition 1. A public-key encryption scheme $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ is *semantically secure* if for all probabilistic polynomial-time algorithms A there exists a probabilistic polynomial-time algorithm B with the following property: for all S, f, h , there is negligible function η such that for all k,

$$\Pr[\texttt{expA}(k) \to 1] - \Pr[\texttt{expB}(k) \to 1] \leq \eta(k).$$

This definition is originally due to [GM84]. There are many variations of it; this particular version follows [Gol04].

Theorem 1 ([GM84]). *A cryptosystem is semantically secure if and only if it is polynomially secure.*

The proof is not nearly as simple as in the information-theoretic case (we will not do it here; see [GM84] for the original proof and [DR98] for a simpler one; [Gol04] has all the details); in fact, the result is surprising to many. There are other definitions of security that turn out to be equivalent to this one, which shows that our understanding of the security of encryption is quite robust.

Semantic security helps prove various cryptographic constructs that use encryption as part of a larger protocol. It can be much more powerful than indistinguishability when used in proofs, because it essentially says that no interesting function of the plaintext can be computed by the adversary.

8.2 Public-key encryption in the real world

Most of encryption that actually happens in daily life is symmetric, not public-key. For example, banks and ATMs rely mostly on the symmetric cipher DES (which we will discuss eventually, but only briefly). Even when public-key encryption is used, it is used only to encrypt a symmetric key, which is then used to encrypt bulk data, because symmetric techniques are much faster than public-key ones.

As far as algorithms used in practice, the most popular one is by far RSA, and the second is ElGamal. Neither is used exactly as we studied it.

In fact, the most common way to use RSA until recently has been a standard known as PKCS #1 version 1.5 [RSA93]. To encrypt a message m, it specifies that one should pad it to the length of the modulus by prepending a zero byte, byte of value 2, at least eight (and as many as needed) random non-zero bytes, followed by another zero byte to separate the pad from the message itself. The resulting bit string gets exponentiated to the public exponent e modulo n.

There is little one can prove about this scheme, although recently Jonsson and Kaliski [JK02] proved its security in certain applications under a relatively strong assumption. At some point it was believed to be not only polynomially secure, but, in fact, secure even against chosen-ciphertext attacks. However, Bleichenbacher [Ble98] found a reasonably practical chosen-ciphertext attack against it. At that time, version 2.0 of PKCS #1 was in the works; currently the most recent version is 2.1. Both 2.0 and 2.1 can be proven not only semantically secure, but also secure against chosen-ciphertext attacks, in a special (unrealistic) model known as "random oracle model." Whether a proof in such a model is actually meaningful is a matter of some debate; we'll consider this subject later in the course. It seems that PKCS encryption is the most common standard used today.

Most problems in implementing encryption, however, do not come from considerations of provability. Rather, they come from we often dismiss as "implementation issues." I identified three of them in class

1. Randomness. Computers, cell phones, ATMs, etc., generally do not come equipped with good sources of random bits that would be unpredictable to the adversary. As we know, though, secret randomness is necessary for key generation and encryption.

2. Secrets. They are hard to keep secret. Today's popular operating systems tend not to provide ways of storing a secret in such a way that it is accessible only to authorized programs and to no one else. A common approach is store a secret encrypted with a password known only to the user. Unfortunately,

users are terrible and remembering high-entropy passwords; in addition, the secret is vulnerable when it's decrypted and actually used in a computation.

3. Keys. As emphasized above, it's very important to authentically know the public key of the person you are sending the message to. There are some approaches to this problem we will discuss later in the course, but they all have drawbacks.

A warning about terminology In the academic world, "public" key and "secret" key usually form a pair. In the commercial world, the name of the second component is often "private" key (which doesn't abbreviate nicely, where as $(\mathrm{PK}, \mathrm{SK})$ does). This wouldn't be too much of a problem, except that the commercial world also often uses "secret key" to mean "non-public key," such as DES, one-time-pad, etc. To avoid confusion, we will call things like DES and the one-time-pad "symmetric" cryptography (because both parties share the same key). (To further compound the confusion, some people use the term "private-key cryptography" to mean "symmetric cryptography".)

8.3 Man-in-the-middle attack against encryption

Note that man-in-the-middle attack also applies to encryption. If Bob wants to send something to Alice, and the two never met before, then Alice needs to send Bob her PK_A. If Eli intercepts it and substitutes his own PK_E instead, Bob won't know the difference. He will now encrypt his message to Alice using PK_E, thus allowing Eli to read it.

In other words, while public-key encryption removes the need to share keys secretly, it does not remove the need for sharing them *authentically*. Bob need not keep PK_A secret, but he does need to know that it came from Alice. We'll address this problem in the next lecture.

References

[Ble98] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *Advances in Cryptology—CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, 23–27 August 1998.

[DR98] Yevgeniy Dodis and Matthias Ruhl. A simple proof that GM-security → semantic security, 1998. Available from theory.lcs.mit.edu/˜yevgen/ps/proof.ps.gz.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.

[JK02] Jakob Jonsson and Burton S. Kaliski, Jr. On the security of RSA encryption in TLS. In Moti Yung, editor, *Advances in Cryptology—CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 127–142. Springer-Verlag, 18–22 August 2002.

[RSA93] PKCS #1: RSA encryption standard. Version 1.5, November 1993. Available from `http://www.rsaisecurity.com/rsalabs/pkcs/`.