CSCI-GA.3210-001 MATH-GA.2170-001	**Introduction to Cryptography** February 29, 2012

Lecture 6

Lecturer: Yevgeniy Dodis *Spring 2012*

1 Public-Key Encryption

Last lecture we studied in great detail the notion of pseudorandom generators (PRG), a deterministic functions that stretch randomness by any *polynomial amount*: from k to $p(k)$ bits. As we already indicated, PRG's have a lot of applications including constructions of both public- and private-key encryptions, and implementation of "ideal randomness" in essentially any programming language. In this lecture we will begin examining these application in more detail by starting with the formal study of public-key encryption (PKE). As we explained before, the informal scenario is this:

- **Before the Encryption.** Alice publishes to the world her public key PK. Therefore, both Bob and Eve know what PK is. This public key is only used to encrypt messages, and a separate key SK is used to decrypt messages. (This is unlike the Secret-Key scheme where one key S is used to both encrypt and decrypt.) Only Alice knows what SK is, and nobody else, not even Bob.
- **Encryption.** When Bob wishes to send Alice a plaintext message M via the Internet, Bob encrypts M using Alice's public key PK to form a ciphertext C. (Formally, we summarize encryption with PK as E_{PK} and say that $C = E_{PK}(M)$.) Bob then sends C over the Internet to Alice.
- **Decryption.** Upon receiving C, Alice uses her secret private key SK to decrypt C, giving her M, the original plaintext message. (Formally, we summarize decryption with SK as D_{SK} and say that $D_{SK}(C) = D_{SK}(E_{PK}(M)) = M$.)
- **Eve's Standpoint.** Unlike the Secret-Key scheme, Eve knows everything Bob knows and can send the same messages Bob can. And, only Alice can decrypt. And, when Bob sends his message, Eve only sees C, and knows PK in advance. But, she has no knowledge of SK. And, if it is hard for Eve to learn about SK or plaintexts based on ciphertexts and PK, then our system is secure.

2 Definition of Public-key Encryption

We start with the syntax of a public-key encryptions scheme, and only later talk about its security. Definition 1 [Public-key encryption (PKE)] A PKE is a triple of PPT algorithms $\mathcal{E} = (Gen, E, D)$ where:

1. *Gen* is the *key-generation algorithm*. $Gen(1^k)$ outputs (PK, SK, M_k), where SK is the secret key, PK is the public-key, and M_k is the message space associated with the PK/SK-pair. Here k is an integer usually called the *security parameter*, which determines the security level we are seeking for (i.e., everybody is polynomial in k and adversary's "advantage" should be negligible in k).

2. E is the *encryption algorithm*. For any $m \in M_k$, E outputs $c \stackrel{r}{\leftarrow} E(m;\ PK)$ — the encryption of m. c is called the *ciphertext*. We sometimes also write $E(m;\ PK)$ as $E_{PK}(m)$, or $E(m;\ r, PK)$ and $E_{PK}(m;\ r)$, when we want to emphasize the randomness r used by E.

3. D is the decryption algorithm. $D(c;\ SK) \stackrel{r}{\rightarrow} \tilde{m} \in \{\mathsf{invalid}\} \cup M_k$ is called the decrypted message. We also sometimes denote $D(c;\ SK)$ as $D_{SK}(c)$, and remark that usually D is deterministic.

4. We require the **correctness** property: if everybody behaves as assumed

$$\forall m \in M_k, \quad \tilde{m} = m, \quad \text{that is} \quad D_{SK}(E_{PK}(m)) = m$$

$\diamondsuit$

Example: RSA. Let us check that RSA satisfies the above definition. Notice, both E and D are deterministic.

1. $Gen(1^k)$ corresponds to the following algorithm: (p, q) are random primes of k bits, $n = pq,\ e \stackrel{r}{\leftarrow} \mathbb{Z}^*_{\varphi(n)},\ d = e^{-1} \bmod \varphi(n),\ M_k = \mathbb{Z}^*_n$. Set $PK = (n, e),\ SK = d$.

2. $c = E(m;\ (n, e)) = m^e \bmod n$.

3. $\tilde{m} = D(c;\ (d, n)) = c^d \bmod n$.

More generally, we could construct a PKE from any TDP. Suppose we have a TDP f with trap-door information t_k and algorithm I for inversion. Here is the induced PKE:

1. $Gen(1^k) \stackrel{r}{\rightarrow} (f, t_k, \{0,1\}^k)$, and f is the PK and the trapdoor t_k is the SK.

2. $E(m;\ PK) = f(m)$.

3. $D(m;\ SK) = I(c, t_k)$.

Conventions about message spaces M_k. Without loss of generality we will assume that the message space M_k can be determined from the public key PK (so we will not explicitly output its desciption in *Gen*). Also, in many schemes the message space M_k does not depend on the particular public key PK and depends only on k, e.g. $M_k = \{0,1\}$ and $M_k = \{0,1\}^k$. In the latter cases we will sometimes say that $\mathcal{E}$ is an encryption for a sequence of message spaces $\{M_k\}$. Notice, however, that in most concrete examples (i.e., in the RSA example above), M_k could depend on the PK. As we will see, this will create some definitional issues when defining security of encryption.

Problems. The problem of the above general construction is that it does not meet our requirement of security.

- First, it reveals partial information. For example, in the RSA case, $f(m)$ preserves the Jacobi symbol of m. Furthermore, if f' is a TDP $f(a,b) = (a, f'(b))$ is also a TDP however it reveals half of the input message.
- Second, our definition of TDP is based on the assumption of uniform distribution of the input. Here, it corresponds to the uniformity of the distribution on the message space. However, in practice, such uniformity is rarely satisfied, and in some interesting cases, the message spaces is actually quite sparse, for example, the English text.
- Third, when the message space is sparse (i.e., sell/buy), this method is completely insecure and can be broken by a simple exhaustive search.
- Many more problems exist. For example, the adversary can tell whether the same message is being sent twice or not.

For completeness, we would like to point out the general construction does satisfy a very weak security notion.

Definition 2 [One-way secure encryption] A PKE $\mathcal{E}$ is called *one-way secure* if it is hard to **completely** decrypt a **random** message. Formally, for any PPT A

$$\Pr(A(c) = m \mid (SK, PK) \stackrel{r}{\leftarrow} Gen(1^k), m \stackrel{r}{\leftarrow} M_k, c \stackrel{r}{\leftarrow} E_{PK}(m)) \leq \mathsf{negl}(k)$$

$\diamondsuit$

Here is a simple lemma directly from the definitions of TDP that shows

Lemma 1 *If $M_k = \{0,1\}^k$ is the domain of a TDP f, then the PKE induced from f is one-way secure.*

Conclusions.

- Much stronger definition is needed in order not to reveal partial information.
- Encryption scheme cannot be deterministic in order to solve the problem of non-uniform/sparse message space.
- Even starting with 1-bit encryption, $M_k = \{0,1\}$, is interesting and non-trivial.

3 Secure encryption of one bit

From the previous section, we discussed the problems with one-way security and the straightforward usage of a TDP. In order to have a stronger definition, let us first begin from from trying to encrypt one bit, i.e. $M_k = \{0,1\}$.

One of the conclusions we had is that the encryption scheme must be probabilistic. For each bit from $M_k = \{0,1\}$, there is cloud of messages in the encrypted message space C corresponding to that bit. Informally, we want the distribution of these two clouds to be

indistinguishable to the adversary *even conditioned on the public key* PK and even though their supports are totally disjoint (the disjointness is from the fact that we want to decrypt the message without ambiguity). We write it as $\langle PK, E_{PK}(0)\rangle \approx \langle PK, E_{PK}(1)\rangle$, where $E_{PK}(0)$ and $E_{PK}(1)$ are two random variables denoting random encryption of 0 and 1 respectively. Notice this is not possible in the Shannon theory, because there the adversary has infinite power and the only way for the distribution to be indistinguishable is that they are exactly the same, which is not the case since the supports of two distributions are totally different. However, in our case it is doable because we assume the adversary is only PPT. We also point out that when the public key PK is clear, we will sometime be sloppy and simply write $E(0) \approx E(1)$, always implicitly assuming that PK is public knowledge. Here is the formal definition.

DEFINITION 3 A PKE for $M_k = \{0,1\}$ is called *polynomially indistinguishable* if $\langle PK, E_{PK}(0)\rangle \approx \langle PK, E_{PK}(1)\rangle$, meaning that $\forall$ PPT A

$$\Big|\ \Pr(A(c, PK) = 1 \mid (SK, PK) \xleftarrow{r} Gen(1^k), c \xleftarrow{r} E_{PK}(0)) - \Pr(A(c, PK) = 1 \mid (SK, PK) \xleftarrow{r} Gen(1^k), c \xleftarrow{r} E_{PK}(1))\ \Big| \ \leq \ \mathsf{negl}(k)$$

Or equivalently,

$$\left|\ \Pr(A(c, PK) = b \mid (SK, PK) \xleftarrow{r} Gen(1^k), b \xleftarrow{r} \{0,1\}, c \xleftarrow{r} E_{PK}(b)) - \frac{1}{2}\ \right| \leq \mathsf{negl}(k)$$

$\diamondsuit$

Example. Suppose f is a TDP with trapdoor information t_k and efficient algorithm I for inversion, and h is a hardcore bit for f. Here is the PKE we informally considered earlier:

1. $Gen(1^k) \xrightarrow{r} (f, t_k)$.
2. $E(b) \to \langle f(x), h(x) \oplus b\rangle = \langle y, d\rangle$. ($x$ is random in $\{0,1\}^k$).
3. $D(\langle y, d\rangle, t_k) :\ x = I(y, t_k), b = d \oplus h(x)$.

Here is another, slightly more efficient suggestion:

1. $Gen(1^k) \xrightarrow{r} (f, t_k)$.
2. $E(b)$: sample $x \xleftarrow{r} \{0,1\}^k$ until $h(x) = b$, then set ciphertext $y = f(x)$.
3. $D(y)$: recover $x \xleftarrow{r} I(y, t_k)$, then decrypt $\tilde{b} = h(x)$.

Notice, E is efficient, since sampling the right x will terminate after approximately two trials, since h must be balanced between 0 and 1. We analyze these schemes formally later (or in the homework).

4 Secure Encryption of Many bits

Now, we will consider the case $M_k = \{0,1\}^{p(k)}$, where p is some polynomial in k. The definition is an obvious generalization of the "bit" version.

Definition 4 [Polynomial indistinguishability] A PKE $\mathcal{E}$ for $M_k = \{0,1\}^{p(k)}$ is called *polynomially indistinguishable* (against PK-only attack) if for any $m_0, m_1 \in M_k$, we have $\langle PK, E_{PK}(m_0)\rangle \approx \langle PK, E_{PK}(m_1)\rangle$. Formally, $\forall$PPT A

$$\left| \Pr(A(c, PK) = b \mid (SK, PK) \stackrel{r}{\leftarrow} Gen(1^k), b \stackrel{r}{\leftarrow} \{0,1\}, c \stackrel{r}{\leftarrow} E_{PK}(m_b)) - \frac{1}{2} \right| \leq \mathsf{negl}(k)$$

$\diamondsuit$

Comments.

- The definition includes the situation when m_0 and m_1 are the same. In this case, no matter $b = 0$ or $b = 1$, E and A will know nothing about what b is, because they only see the message m_b, which is the same, no matter $b = 0$ or $b = 1$.
- As will will see this definition is extremely robust and prevents a lot of attacks. For example, it also excludes the possibility for the adversary to tell whether a message was being sent twice. Informally, if A could determine this, when given c, A can generate $c' \leftarrow E(m_0)$, and see if c and c' correspond to the same message, thus determining if $b = 0$.

Blum-Goldwasser construction. In the Blum-Goldwasser construction, as we mentioned earlier, we are given a TDP f with trapdoor t_k, inversion algorithm I, and a hardcore bit h. Recall also that if we let $G(x) = G'(x) \circ f^{(n)}(x)$, where $G'(x) = h(x) \circ h(f^1(x)) \circ \cdots \circ h(f^{(n-1)}(x))$, then both G and G' are PRG's. We define:

1. $PK = f$ and $SK = t_k$.
2. $E(m)$: get $x \stackrel{r}{\leftarrow} \{0,1\}^k$, send $c = (G'(x) \oplus m, f^{(n)}(x))$.
3. $D(c)$: use t_k to get $f^{(n-1)}(x), \ldots, f(x), x$, and use them to calculate $G'(x)$ with hardcore bit function h. After we have $G'(x)$, recovering m is clear.

To check the correctness of Blum-Goldwasser construction, we need to prove that for all m_0 and m_1 (below we omit $PK = f$ since it's fixed)

$$E(m_0) \equiv (f^{(n)}(x), G'(x) \oplus m_0) \approx (f^{(n)}(x), G'(x) \oplus m_1) \equiv E(m_1) \tag{1}$$

In order to prove this, we will instead prove a more general lemma.

Lemma 2 (One-Time Pad Lemma) *Let R denote the uniform distribution. Then for all distributions X, Y (not necessarily independent!), if $(X, Y) \approx (X, R)$, then for all m_0 and m_1 we have $(X, Y \oplus m_0) \approx (X, Y \oplus m_1)$.*

Proof: The simplest proof is to notice that for any fixed message m, $R \oplus m \equiv R$, where R is a random string. I.e., "random+fixed=random". Thus, since XOR is an efficient operation,

$$(X, Y \oplus m_0) \approx (X, R \oplus m_0) \equiv (X, R) \equiv (X, R \oplus m_1) \approx (X, Y \oplus m_1)$$

□

We see that Lemma 2 indeed implies the needed Equation (1). Indeed, consider $X = f^{(n)}(x)$, $Y = G'(x)$. Then, to apply Lemma 2, we only need to argue that $(f^{(n)}(x), G'(x)) \approx (f^{(n)}(x), R')$, where R' is random. But since f is a permutation and x is random, $f^{(n)}(x)$ is just a random string, so $(f^{(n)}(x), R') \equiv R$, and we just need to show that $G(x) = f^{(n)}(x) \circ G'(x)$ is a PRG, which is precisely what we showed last time. Thus, we get

Theorem 1 *BG construction above defines a polynomially indistinguishable encryption.*

As a special case, we also get the security of the one-bit version of BG encryption that we considered in the previous section.

Efficient example: squaring over Blum integers. Recall, the Blum-Blum-Shub construction of G' uses the OWF $SQ(x) = x^2 \bmod n$. This function is a TDP when $n = pq$ with $p = 3 \bmod 4$ and $q = 3 \bmod 4$. Specifically, it can be proved that it is a permutation on SQ_n, and the trapdoor key is the factorization (p, q) of $n = pq$. The associated hardcore bit is the least significant bit of x. Now we see that this construction is quite efficient, because in order to encrypt $p(k)$ bits, we only need to do $p(k)$ multiplications mod n.

5 Key Encapsulation Mechanism

The BG example above follows the following *key encapsulation principle* which we will meet in virtually any public-key encryption scheme. The idea is to derive a "radom-looking" key s and its "encryption", and then use s to "one-time pad the message". For example, in the BG scheme above, the key s was equal to $G'(x)$, while the "encryption" of s was the value $\psi = f^{(n)}(x)$. A bit more formally,

Definition 5 [Key Encapsulation Mechanism (KEM)] A *Key Encapsulation Mechanism* is a triple of PPT algorithms $\mathcal{E} = (Gen, KE, KD)$ where:

1. *Gen* is the *key-generation algorithm*. $Gen(1^k)$ outputs (PK, SK, M_k), where SK is the secret key, PK is the public-key, and M_k is the "key space" associated with the PK/SK-pair.

2. KE is the *key encapsulation algorithm*. It takes the public key PK and outputs a pair $\langle \psi, s \rangle \stackrel{r}{\leftarrow} KE(PK)$, where $s \in M_k$ is a key and ψ is called the *ciphertext* representing encryption of s. We sometimes write $\langle \psi, s \rangle = KE_{PK}(r)$ to emphasize the randomness r used by KE.

3. KD is the key decapsulation algorithm. $KD(\psi;\ SK) \stackrel{r}{\rightarrow} \tilde{s} \in \{\mathsf{invalid}\} \cup M_k$ attempts to extract a key from the ciphertext ψ. We sometimes denote $KD(\psi;\ SK)$ as $KD_{SK}(\psi)$, and remark that usually KD is deterministic.

4. We require the **correctness** property: if $\langle \psi, s \rangle \stackrel{r}{\leftarrow} KE(PK)$, then $KD(\psi, SK) = s$.

$\diamondsuit$

As we mentioned, in the BG example, $KE(x)$ uses random x to set $s = G'(x)$, $\psi = f^{(n)}(x)$, while $KD(\psi) = G'(f^{(-n)}(\psi))$.

The usefulness of KEM comes from the fact that it immediately yields a PKE, by using the symmetric key s to encrypt the message m. For concreteness, below we assume $M_k = \{0,1\}^n$ for some parameter $n = p(k)$ (like in the BG case), and we will use the one-time pad encryption to encrypt m (although later we will see that any symmetric-key encryption will do!). But first we need a definition of security for KEM.

DEFINITION 6 [Polynomial indistinguishability] A KEM (Gen, KE, KD) for $M_k = \{0,1\}^{p(k)}$ is called *polynomially indistinguishable* (against PK-only attack) if for randomly generated PK and $\langle \psi, s \rangle \leftarrow KE(PK)$ we have $\langle PK, \psi, s \rangle \approx \langle PK, \psi, R \rangle$, where R is a fresh random string sampled from M_k. Formally, $\forall$PPT A

$$\begin{aligned} \Pr(A(\psi, s_b, PK) = b \quad | \quad & (SK, PK) \stackrel{r}{\leftarrow} Gen(1^k), b \stackrel{r}{\leftarrow} \{0,1\}, \\ & (\psi, s_0) \stackrel{r}{\leftarrow} KE(PK), s_1 \stackrel{r}{\leftarrow} \{0,1\}^{p(k)}) \\ & \leq \frac{1}{2} + \mathsf{negl}(k) \end{aligned}$$

$\diamondsuit$

Lemma 3 *Assume (Gen, KE, KD) is polynomially indistinguishable* KEM *for $\{0,1\}^n$. Then the following* PKE *is polynomially indistinguishable for $\{0,1\}^n$:*

- *key generation Gen is the same as in* KEM.
- *encryption $E(m)$: compute $\langle \psi, s \rangle \leftarrow KE$, and let $c = \langle \psi, m \oplus s \rangle$.*
- *decryption $D(\psi, z) = z \oplus KD(\psi)$.*

Proof: The proof immediately follows from the One-Time Pad Lemma, where $X = (PK, \psi)$, and $Y = s$. $\square$

We will see several other applications of the KEM paradigm. In particular, we will see that the one-time pad can be replaced by any (one-time) secure symmetrci-key encryption. As a concrete illustration, we can apply it to the symmetric scheme $E_s(m) = G(s) \oplus m$, where G is any PRG. Although we did not yet prove that this scheme is secure, we directly prove that it yields a good KEM.

Lemma 4 *Assume (Gen, KE, KD) is polynomially indistinguishable* KEM *for $\{0,1\}^k$ and G is a* PRG *from $\{0,1\}^k$ to $\{0,1\}^n$. Then the following (Gen', KE', KD') is a polynomially indistinguishable* KEM *for $\{0,1\}^n$:*

- *$Gen' = Gen$.*
- *$KE'(PK)$: compute $\langle \psi, s \rangle \leftarrow KE(PK)$, and let $\psi' = \psi$, $s' = G(s)$. Output $\langle \psi', s' \rangle$.*

- $KD'(\psi') = G(KD(\psi'))$.

We leave the proof as a simple exercise in the hybrid argument. We also remark that a trivial generalization of the above result shows that *any* polynomially indistinguishable encryption (Gen, E, D) on $\{0,1\}^k$ can be combined with any secure PRG from k to n bits to directly give a polynomially indistinguishable encryption (Gen, E', D') on $\{0,1\}^n$: simply pick a random k-bit key s, encrypt it using E, and append $G(s) \oplus m$ to obtain the encryption of m. This shows that one can, in principle, only show how to encrypt relatively short messages, and then be able to encrypt much longer messages, provided a good PRG is availble.

6 General transformation from one bit to many bit

Blum-Goldwasser construction shows that given a TDP, we could transfer many bits, by efficiently generalizing the corresponding original BG scheme to encrypt one bit. More generally, the above discussion following Lemma 4 shows that we only need to encrypt messages roughly as long as the security parameter to be able to encrypt much longer messages. However, suppose we have some PKE scheme for one bit, which possibly does not depend on any TDP and does not seem to obviously generalize to encrypt many bits. In fact, assume that the only thing we know about it is that it is indistinguishable one-bit encryption. Is it possible for us to use this scheme to encrypt many bits without using any other assumptions on this scheme? A naive answer is to regard each bit to be a separate message and encrypt it using the PKE scheme for one bit. Luckily, this naive approach works for *public* key encryption.[1] Formally, let $\mathcal{E} = (Gen, E, D)$ be a polynomial indistinguishable PKE scheme for one bit, we could define a PKE scheme $\mathcal{E}' = (Gen', E', D')$ for $M_k = \{0,1\}^n$ ($n = p(k)$ for some polynomial p) as follows:

1. $Gen'(1^k) = Gen(1^k) \to (PK, SK)$, i.e. PK and SK are generated in the same way as before Gen, except the message space now is $M_k = \{0,1\}^{p(k)}$. Now, given $m \in M_k = \{0,1\}^n$, we denote m as $m^1 m^2 \cdots m^n$.
2. Define $E'_{PK}(m^1 m^2 \cdots m^n) = (E_{PK}(m^1), E_{PK}(m^2), \cdots, E_{PK}(m^n)) \to c^1 c^2 \cdots c^n = c$.
3. Define $\tilde{m} = D'_{SK}(c^1 c^2 \cdots c^n) = (D_{SK}(c^1), D_{SK}(c^2), \cdots, D_{SK}(c^n))$

And it turns out that this bit-by-bit encryption indeed works!

Theorem 2 *If $\mathcal{E}$ is polynomially indistinguishable for one bit, then $\mathcal{E}'$ is polynomial indistinguishable for $n = p(k)$ bits.*

Proof: Take two messages m_0 and m_1. We first construct a sequence of intermediate messages that slowly go from m_0 to m_1:

$$\begin{array}{cccccccc}
M_0 & = & m_0^1 & m_0^2 & \dots & m_0^{n-1} & m_0^n \\
M_1 & = & m_1^1 & m_0^2 & \dots & m_0^{n-1} & m_0^n \\
\vdots & & & & \vdots & & \\
M_{n-1} & = & m_1^1 & m_1^2 & \dots & m_1^{n-1} & m_0^n \\
M_n & = & m_1^1 & m_1^2 & \dots & m_1^{n-1} & m_1^n
\end{array}$$

[1] As we will see, things are a bit more complex in the symmetric key setting.

Notice, $M_0 = m_0$ and $M_n = m_1$. Also, M_{i-1} and M_i differ in at most one bit — bit number i. We now define a sequence of distributions

$$C_i \leftarrow E'(M_i) = E(m_1^1) \dots E(m_1^i) E(m_0^{i+1}) \dots E(m_0^n)$$

Using the hybrid argument, in order to prove that (we omit the PK from all the distributions for compactness)

$$E'(m_0) = E'(m_0^1 m_0^2 \cdots m_0^n) \approx E'(m_1^1 m_1^2 \cdots m_1^n) = E'(m_1)$$

i.e. $C_0 \approx C_n$, we only need to show that for any i, we have $C_{i-1} = E'(x_{i-1}) \approx E'(x_i) = C_i$. Graphically,

$$\begin{array}{rcccccccc} C_{i-1} & = & E(m_1^1) & \dots & E(m_1^{i-1}) & \boxed{E(m_0^i)} & E(m_0^{i+1}) & \dots & E(m_0^n) \\ C_i & = & E(m_1^1) & \dots & E(m_1^{i-1}) & \boxed{E(m_1^i)} & E(m_0^{i+1}) & \dots & E(m_0^n) \end{array}$$

Now, let $A = E(m_1^1) \circ \dots \circ E(m_1^{i-1})$, $B = E(m_0^{i+1}) \circ \dots \circ E(m_0^n)$. Thus, we only need to show that

$$(PK, E(m_0^i), A, B) \approx (PK, E(m_1^i), A, B)$$

But this is obvious now! Since by our assumption on (Gen, E, D), we have $(PK, E_{PK}(m_0^i)) \approx (PK, E(m_1^i))$, and since both A and B *can be computed in polynomial time with the knowledge of* PK (i.e., $(A, B) = g(PK)$ for some *efficient* function g), we immediately get the desired conclusion. □

To recap the whole proof, we used the fact that if one can distinguish between encryptions of two long messages encrypted bit-by-bit, there must be some particular index i that gives the adversary this advantage, but this contradicts the bit security of our base encryption scheme. Also, notice that we loose a polynomial factor $p(k) = n$ in security by using the hybrid argument, but this is OK since n is polynomial.

Remark 1 *We notice that the above one-bit to many-bit result is false for private-key encryption (which we did not cover formally yet). For example, consider the one-time pad with secret bit s and $E_s(b) = b \oplus s$. We know it is perfectly secure. However, if we are to encrypt two bits using s, $c_1 = b_1 \oplus s$ and $c_2 = b_2 \oplus s$, then $c_1 \oplus c_2 = b_1 \oplus b_2$, so we leak information. The key feauture of the public-key encryption that makes the result true is the fact that anyone can encrypt using the public key, which is false in the private-key case (check that this is the place where the proof fails).*

Remark 2 *Although we stated the result for one-bit to many-bits, it is clearly more general. In particular, if we have an indistinguishable encryption of b bits, then we easily get an indistinguishable encryption of bn bits, for any $n = \mathsf{poly}(k)$, by simply splitting the message into n chucks of b bits each, ach encrypting each chunk separately.*

7 Chosen Plaintext Secuirty

We have been careful so far to only state the notion of polynomial indistinguishability for the message space $\{0,1\}^n$. However, most practical encryption scheme have general message spaces, which usually depends on the specific public key we choose at key generation. Initially, it seems like it is trivial to generalize our definition for $\{0,1\}^n$ to general message spaces. However, we will see that the resulting notion is slightly incorrect. In fact, it suffers from both a syntactic criticism and a semantic problem.

Syntactic Criticism. A first issue with this definition of security is that it requires that, for each pair of messages (m_0, m_1), their encryptions are indistinguishable from each other: $\boxed{(\forall m_0, m_1 \in M_k)}$, $\forall$ PPT algorithm A

$$\left| \Pr[A(c, PK) = b \mid \boxed{(SK, PK, M_k) \leftarrow_r G(1^k)}, b \leftarrow_r \{0,1\}, c \leftarrow_r E_{PK}(m_b)] - \frac{1}{2} \right| \leq \mathsf{negl}(k)$$

In other words, we are quantifying on the message space M_k, even before the random choice of (SK, PK, M_k)! In the cases in which the message space M_k is fixed, or just varies as a function of the security parameter k (e.g. when $M_k = \{0,1\}$ or $M_k = \{0,1\}^{p(k)}$), this can be fixed simply "moving M_k out" of the output of the key-generation algorithm $G(1^k)$. And this is exactly what we did. But in other interesting cases, like the RSA and the ElGamal PKEs we study later, the message space M_k is indeed somehow related to the public key being used, so that it doesn't yet make sense to choose a pair of messages before knowing the actual public key PK being selected.

This is clearly just a syntactic problem, so that it doesn't affect too much the overall correctness of our current approach. Still it needs to be fixed.

Semantic Criticism. A semantic problem with the notion of indistinguishable security is that it just states that any PPT adversary must have a negligible advantage in distinguishing the encryptions of any pair of messages (m_0, m_1), when the public key is *randomly* selected. Therefore, it does not cast away the chance that, given knowledge of the public key, an adversary may be able to come up with a pair of messages such that it can indeed distinguish the associated ciphertexts.

To overcome this problem, we could be tempted to modify the definition in such a way that first we fix the public key PK and the private key SK, and then we quantify overall the possible pair of messages. But in this case we are asking too much, since the existence of "bad" pairs of messages is unavoidable. To see why, consider an adversary that, in distinguishing between the encryptions of (m_0, m_1), always assumes that the second message is the private key SK, and tries to decrypt the encryption it has being given accordingly. Clearly, the advantage of this adversary is negligible whenever the second message is not the secret key; however, in the very specific, pathological case in which the pair is of the form (m_0, SK), its advantage will be 1.

After all, what we really want from our definition of security is that, even if the adversary already knows PK, it should be infeasible for him to find two messages for which it has a non-negligible advantage.

Security Against Chosen Plaintext Attack. Both the criticisms stated above stem from the the fact that the messages m_0, m_1 are quantified "outside" the probability which constitutes the advantage of the adversary:

- from a syntactic point of view, this is bad since the message space M_k is chosen "inside" the probability;
- from a semantic point of view, this is bad since we want to allow the adversary to choose the pair of messages after seeing the PK, and thus the probability should be taken over all the pairs of messages that it can efficiently find.

We can fix both this problems with the following definition of:

Definition 7 [Indistinguishable Security Against Chosen Plaintext Attack]
A PKE $\mathcal{E} = (G, E, D)$ is *indistinguishable against a chosen plaintext attack (*CPA*)*, or, shortly, IND-CPA-*secure*, if $\forall$ PPT algorithm A

$$\left| \Pr\left[b = \tilde{b} \;\middle|\; \begin{array}{c} (PK, SK, M_k) \leftarrow_r G(1^k), (m_0, m_1, \alpha) \leftarrow A(PK, M_k, \text{'find'}) \\ b \leftarrow_r \{0,1\}, c \leftarrow E_{PK}(m_b), \tilde{b} \leftarrow A(c, \alpha, \text{'guess'}) \end{array} \right] - \frac{1}{2} \right| = \mathsf{negl}(k)$$

$\diamondsuit$

Let's take a closer look at what is going on! This notion of security can be viewed as the following game between us and the adversary:

1. we run the key-generation algorithm, obtaining (PK, SK, M_k);
2. we give the public key PK and the message space M_k to the adversary and ask it to 'find' a pair of messages in M_k for which it believes it can distinguish encryptions under the key PK;
3. the adversary outputs a pair of messages (m_0, m_1) of its choice, along with its final "state", i.e. some kind of summary of the considerations and computations that led it to choose this specific pair of messages;
4. we choose which one of two message we want encrypt, and give it the corresponding ciphertext c;
5. we ask the adversary to tell which message we encrypted, allowing it to "remember" the reasons for which it chosen the pair (m_0, m_1).

We win the game (i.e. the considered PKE is IND secure against CPA) if the adversary's advantage (defined as usual as its probability of success) is negligible.

As we will see, the resulting notion is really the "right" notion of security for PKE. Moreover, it turns out it is better suited even for the case when the message space is $\{0,1\}^n$. Namely, we show that this new notion of security is strictly more general than the PK-only security, even when restricted to $\{0,1\}^n$ for which PK-only security also makes sense.

Theorem 3 (CPA $\Rightarrow$ PK-only)
If a PKE $\mathcal{E} = (G, E, D)$ *is* CPA *secure, then* $\mathcal{E}$ *is also* PK*-only secure.*

Proof: Let us assume that $\mathcal{E}$ in not PK-only secure, i.e.
$\exists m_0, m_1$, $\exists$ a PPT algorithm A such that

$$\Pr(A(c, PK) = 1 \mid (SK, PK) \leftarrow_r G(1^k), c \leftarrow_r E_{PK}(c)) = \epsilon$$

where ϵ is not-negligible.

Now, we can construct an adversary A' that, using A as a black-box module, breaks the CPA security of $\mathcal{E}$. Recall that, in the CPA notion of security, the adversary acts in two rounds:

find Regardless of the public key PK at hand, A' always chooses the pair of message (m_0, m_1) of the hypothesis, and records in α the public key PK, m_0 and m_1:

$$A'(PK, M_k, \text{`find'}) \rightarrow (m_0, m_1, \alpha)$$

guess When challenged to determine which message was encrypted, A' runs $A(c, PK)$:

$$A'(c, \alpha, \text{`guess'}) \rightarrow A(c, PK)$$

Clearly, this algorithm A' has the same (non-negligible) advantage as A, contradicting the hypothesis that $\mathcal{E}$ was CPA secure. □

Theorem 4 (PK-only $\not\Rightarrow$ CPA)
There exists a PKE $\mathcal{E} = (G, E, D)$ *which is* PK*-only secure, but is not* CPA *secure.*

Proof: To prove the theorem it suffices to come up with a counterexample, namely a specific PKE $\mathcal{E}'$ which is PK-only secure but not CPA secure. To this purpose, consider an arbitrary PKE $\mathcal{E} = (G, E, D)$ which is PK-only secure and modify it as follow:

Key-Generation Algorithm

$$\begin{aligned} &\textbf{set:} && (PK, SK) = G(1^k), r \text{ random} \\ &\textbf{output:} && (PK', SK') = ((PK, r), SK) \end{aligned}$$

Encryption

$$E'(m) = \begin{cases} (0, E(PK')), & \text{if } m = PK' \\ (1, E(m)), & \text{otherwise} \end{cases}$$

Decryption

$$\begin{cases} D'(0, y) = PK' \\ D'(1, y) = D(y) \end{cases}$$

Clearly, $\mathcal{E}'$ is *not* CPA secure, since once the adversary knows the public key PK', it can ask to be challenged on the pair of messages (PK', m_1), and it will always succeed in distinguishing an encryption of PK' (which starts with 0) from an encryption of any other message (which starts with 1).

Nevertheless, $\mathcal{E}'$ is still a PK-only secure PKE, since, in this setting, it is no longer possible to efficiently find a pair of messages (m_0, m_1) whose encryptions are easy-to-distinguish. Indeed, due to the random r present in the public key PK', the probability that the adversary could guess the public key PK' that will be used, is always negligible (no matter how short the public key PK for the PKE $\mathcal{E}$ could be). Besides, it will be difficult for the adversary to find a generic pair (m_0, m_1) which makes its task easy, since such a pair could be also used to tell apart encryptions for the PKE $\mathcal{E}$, contradicting the hypothesis that $\mathcal{E}$ is PK-only secure. □

Remark 3 *Despite the slight differences, and the non-equivalence proved in the above theorem, the two notions of security formalized are quite close. In particular, all the results stated so far (the generalization of a one-bit secure* PKE *to a secure* PKE *for any number of bits, the Blum-Goldwasser system, the analysis of security of the ElGamal cryptosystem) hold as well when we consider the indistinguishable security against* CPA*.*

This is because the separation between CPA *and* PK*-only security is somewhat artificial, and not of real concern, but anyway, once you know, why not to use the right one?*

8 Semantic Security

We defined the notion of indistinguishability for PKE. For convenience, we repeat it here in a slightly modified (but equivalent form), where we "split" the attacker B into B_1 (corresponding to B in the 'find' stage) and B_2 (corresponding to B in the 'guess' stage)"

Definition 8 [Indistinguishability] A cryptosystem (G, E, D) is called IND-*secure* (against CPA attack) if for any PPT adversary $B = (B_1, B_2)$, we have

$$\Pr[\, b = \tilde{b} \;\mid\; \begin{array}{l}(PK, SK) \leftarrow G(1^k);\\ (m_0, m_1, \beta) \leftarrow B_1(PK);\\ b \leftarrow \{0,1\};\\ \tilde{c} \leftarrow E(m_b;\ PK);\\ \tilde{b} \leftarrow B_2(\tilde{c}, \beta, PK)\ ]\end{array} \qquad \leq \tfrac{1}{2} + \mathsf{negl}(k)$$

♢

The notion of indistingusihability is pretty simple, but it is not clear if it is enough for all applications of encryption. Why are there two messages only, and why is the message chosen to be one of them with probability precisely 1/2? And does it imply "security" when message is chosen from a higher entropy distribution? Motivated by these questions, we now define a seemingly much more powerful definition of *semantic security*. It is more difficult to grasp, but it literally says that the knowledge of the ciphertext cannot help the adversary.

Definition 9 [Semantic security] A cryptosystem (G, E, D) is *semantically secure* (against CPA attack) if for any PPT environment Env and any PPT adversary A there exists a PPT simulator S such that $|p_a(k) - p_s(k)| = \mathsf{negl}(k)$, where

$$
\begin{aligned}
p_a(k) \quad = \quad & \Pr[\, R(m,z)=1 \;\mid\; (PK,SK) \leftarrow G(1^k); \\
& \qquad\qquad\quad\; (m,R,\alpha) \leftarrow \mathsf{Env}(PK) \\
& \qquad\qquad\quad\; c \leftarrow E(m;\ PK); \\
& \qquad\qquad\quad\; z \leftarrow A(c,\alpha,PK)\ ]; \\
\\
p_s(k) \quad = \quad & \Pr[\, R(m,z)=1 \;\mid\; (PK,SK) \leftarrow G(1^k); \\
& \qquad\qquad\quad\; (m,R,\alpha) \leftarrow \mathsf{Env}(PK) \\
& \qquad\qquad\quad\; z \leftarrow S(\alpha,PK)\ ].
\end{aligned}
$$

$\diamondsuit$

Let us trace the meaning behind this definition. First, we explain the constructs used in the definition.

Env stands for the environment/context in which the sender, the receiver, and the adversary operate. In the environment, an event may occur that will cause the sender to need to send a certain message to the receiver. E.g. changes in the enemy's strategies may create the need of two army divisions to communicate. When such an event occurs, the environment will generate the message m the sender will have to send and some partial information α the adversary will infer from knowing the event. Notice, such environment could be arbitrary (albeit PPT), so we quantify over all such environments.

The adversary A, as usual, attempts to gather some (more) information about the message from the ciphertext and the partial information α. This information is referred to as $z \in Z$. The adversary is considered successful if his algorithm A outputs information z that is relevant, i.e. it has to do with the message m. In other words, the message m and the adversary's output z must be related. Formally this means that there is a binary relation $R \subset M_k \times Z$ such A tries to satisfy $R(m,z)$. The environment produces this relation R in addition to the message and information α (since Env is arbitrary, this gives a lot of freedom in generating R that the adversary happens to be "interested in").

In the real world (as opposed to the simulated experiment described below), the sender encrypts $c \leftarrow E_{PK}(m)$, and the adversary intercepts c. The adversary has access to c, PK, and the public information α he learns from the environment. The adversary runs his algorithm that outputs z, the information that the adversary hopes would reveal something from the relevant to the message m. The probability of the adversary's success is $p_a = \Pr[R(m,z)=1]$, i.e. the probability that the output information z is in relation R with the message m.

Now let us discuss the simulated experiment. A simulator S is an algorithm very similar to the one the adversary uses (A): S's objective and output is the information z as related to the original message as possible. The only difference is that *S does not receive c as its input*. The simulator operates purely on the public knowledge α. Given a paraticular simulator S, its probability of success is $p_s(k) = \Pr(R(m,z)=1)$.

Now, what we want to say is that no PPT adversary A can benefit too much from intercepting the ciphertext c. Namely, for any such adversary A where exists a simulator S, whcih achieves essentially the same probability of success, without any knowledge of c! In other words, the knowledge of c is useless for all practical purposes: *whatever could be inferred from it, could be inferred without it as well*. Formally, $|p_a(k) - p_s(k)| = \mathsf{negl}(k)$. This is exactly what the definition of semantic security states.

Quite remarkably, we will show that *semantic security is equivalent to indistinguishability*! We notice that by proving this equivalence we will gain the powerful implications of semantic security while being able to work with the simpler IND-security definition. And this is exactly what we show:

Theorem 5 *A cryptosystem (G, E, D) is semantically secure if and only if it is* IND*-secure.*

8.1 IND-security implies semantic security

Suppose a cryptosystem (G, E, D) is not semantically secure. We wish to show that it is not IND-secure either.

By assumption, there exists a PPT environment Env and adversary A such that for all PPT simulators S, $|p_a(k) - p_s(k)| \geq \epsilon$ where ϵ is non-negligible (i.e., it is inverse-polynomial for infinitely many k). So, in particular, let S be as follows:

- S: On input (α, PK),
  - Let $c' \leftarrow E(0;\ PK)$.
    Here by 0 we denote an arbitrary fixed message in the domain of our encryption (we assume wlog that such fixed message can be easily obtained from PK).
  - Output $z' \leftarrow A(c', \alpha, PK)$.

By assumption, for this S we have $|p_a(k) - p_s(k)| \geq \epsilon$. To understand the above better, we see that we really perform the following experiment:

$$\begin{array}{rcl} (PK, SK) & \leftarrow & G(1^k) \\ (m, R, \alpha) & \leftarrow & \mathsf{Env}(PK) \\ c \leftarrow E(m;\ PK) \quad \text{and} & & c' \leftarrow E(0;\ PK) \\ z \leftarrow A(c, \alpha, PK) \quad \text{and} & & z' \leftarrow A(c', \alpha, PK) \end{array}$$

Then $p_a(k) = \Pr(R(m, z) = 1)$, while $p_s(k) = \Pr(R(m, z') = 1)$, and our assumption says

$$|\Pr(R(m, z) = 1) - \Pr(R(m, z') = 1)| \geq \epsilon \tag{2}$$

We notice that the above almost literally defines a distinguisher between encryption of m and of 0. To formalize this, we now construct the following PPT adversary $B = (B_1, B_2)$ that will break the IND-security of our encryption by making sub-routine calls to Env and A:

- B_1: On input PK,
  - Obtain $(m, R, \alpha) \leftarrow \mathsf{Env}(PK)$.
  - Output $(m, 0, \beta)$, where state information $\beta = (m, R, \alpha)$.
- B_2: On input $(\tilde{c}, \beta, PK)$, where $\beta = (m, R, \alpha)$,
  - Obtain $\tilde{z} \leftarrow A(\tilde{c}, \alpha, PK)$.

– If $R(m, \tilde{z}) = 1$, output $\tilde{b} = 0$ (i.e. message is m), else $\tilde{b} = 1$ (i.e. message is 0).

The fact the $B = (B_1, B_2)$ break the indistinguishability of (G, E, D) is almost obvious. Indeed, if $b = 0$ (i.e., message is m), then $\tilde{c} = c \leftarrow E(m;\ PK)$, $\tilde{z} = z \leftarrow A(c, \alpha, PK)$, and $\Pr(\tilde{b} = b) = \Pr(R(m, z) = 1) = p_a(k)$. Similarly, if $b = 1$ (i.e., message is 0), then $\tilde{c} = c' \leftarrow E(0;\ PK)$, $\tilde{z} = z' \leftarrow A(c', \alpha, PK)$, and $\Pr(\tilde{b} = b) = \Pr(R(m, z) = 0) = 1 - p_s(k)$. Overall

$$\Pr(\tilde{b} = b) = \frac{1}{2} \cdot (p_a(k) + 1 - p_s(k)) = \frac{1}{2} + \frac{1}{2} \cdot (p_a(k) - p_s(k))$$

which is non-negligibly different from 1/2 by Equation (2). This completes the proof that IND-security implies semantic security.

8.2 Semantic security implies IND-security

Suppose that our cryptosystem (G, E, D) is not IND-secure, i.e. there exists an adversary $B = (B_1, B_2)$ such that for some non-negligible $\epsilon = \epsilon(k)$,

$$\begin{array}{lll} \Pr[\ b = \tilde{b} \ \mid & (PK, SK) \leftarrow G(1^k); & \\ & (m_0, m_1, \beta) \leftarrow B_1(PK); & \\ & b \leftarrow \{0, 1\}; & \\ & \tilde{c} \leftarrow E(m_b;\ PK); & \\ & \tilde{b} \leftarrow B_2(\tilde{c}, \beta)\ ] & > \frac{1}{2} + \epsilon(k) \end{array}$$

We notice that wlog we can assume that B_1 never outputs $m_0 = m_1$. Let us construct an environemnt Env and an adversary A that break the semantic security of (G, E, D) as follows:

- Env: On input PK,
  - Obtain $(m_0, m_1, \beta) \leftarrow B_1(PK)$.
  - Let R be the equality relation ($R(x, y) = 1 \iff x = y$), and let adversary's partial information be $\alpha = (m_0, m_1, \beta)$.
  - Output (m_b, R, α), where $b \leftarrow \{0, 1\}$ is a random bit (not given to the adversary!)
- A: On input (c, α, PK), where $\alpha = (m_0, m_1, \beta)$,
  - Obtain $\tilde{b} \leftarrow B_2(c, \beta)$.
  - Output $z = m_{\tilde{b}}$.

It is clear that the value $p_a(k)$ for this adversary is exactly the probability of success of $B = (B_1, B_2)$, since the experiments and the success condition ($m_b = m_{\tilde{b}} \iff b = \tilde{b}$ as we assumed $m_0 \neq m_1$) are *exactly the same*. Namely, our environment picked the random bit b for the adversary, but did not reveal it in the partial information α. Thus, $p_a(k) > \frac{1}{2} + \epsilon$.

On the other hand, what is the best chances for any algorithm S that does not take c as input? Well, S has to predict a random bit b about which it gets no information at all (remember that Env picked b after α — the input to S — was already computed)! Hence, $p_s(k) = \frac{1}{2}$, which combined with $p_a(k) > \frac{1}{2} + \epsilon$ contradicts the semantic security of (G, E, D).

8.3 Other definitions

Twenty years ago, there were no established definitions of security. But people had intuition that "seemed right." The intuition behind the RSA cryptosystem is that, without any a-priori information on the message, it is hard to infer it completely from seeing its encrypted form. As we have stressed before, this intuition of "one-wayness" is much weaker than the intuition behind IND-security.

Yao had another intuition, which had to do with effectiveness of data compression. His intuition is that, if a cryptosystem is secure, then it takes just as many bits to transmit a message, whether I happen to have this message in encrypted form already or not. Namely, the encryption does not help me to further compress the message. From the point of view of an information-theorist, this intuition is very natural, and quite akin to Shannon's ideas about security. It turns out that Yao's definition is equivalent to semantic security (and thus indistinguishability)! This was shown by Micali, Rackoff, and Sloan. Hence, we start to see very convincing evidences that *there is something fundamental in our notion of security, since so many differently looking definitions turned out to be equivalent!*

Notice, it is possible that other notions of security may prove useful, even if they are not comparable to existing ones. In particular, semantic security is not the only remaining candidate (even though it is the most accepted one). The study of useful notions of security is on-going, and if you have any interesting ideas related to it, please be sure to follow up on them!

WORRY ABOUT DEFINITIONS SO MUCH? We focus on *provable* security, i.e., if we claim that a cryptosystem is secure, then we also exhibit a proof of this claim, under some reasonable computational assumption (where "reasonable" is in the eye of the beholder, of course). Historically, there were so many incorrect cryptosystems and definitions of security suggested, that the need for formalism is clear: why to settle for empirical evidence when you can do better? Of course, whether a given formal definition satisfies one's need for security, or comes short of it, is, needless to say, in the eye of the beholder as well. To summarize, a *discussion* of possible definitions and their strength is central, while the *need* for such definitions is obvious.