CSCI-GA.3210-001 MATH-GA.2170-001	**Introduction to Cryptography**	March 2, 2012

Lecture 7

Lecturer: Yevgeniy Dodis *Spring 2012*

Last time we proved that if there exists a secure PKE $\mathcal{E}$ to encrypt one bit, then there also exists a secure PKE $\mathcal{E}'$ to encrypt as many bits as we want. In this lecture, we will go further in this direction, searching for the minimal assumption that implies the existence of a indistinguishable (IND) PKE.

Afterwards, we will temporarily leave the construction of our theoretical framework, and we will move to analyzing efficient and practical encryption schemes, based on less general assumptions. In particular, we will present the *ElGamal* PKE, whose various levels of security can be stated in term of two different (but related) assumptions: the *Computational Diffie-Hellman* (CDH) Assumption, and the *Decisional Diffie-Hellman* (DDH) Assumption. We will also study the related problem of *Diffie-Hellman* key exchange.

Finally, we come back to Secret-Key Encryption (SKE) and study notions of security for SKE. First, we define IND-security for one message and give examples of encryptions that provide such security. Unfortunately, one-message security no longer implies multiple message security for SKE. Thus, we define the notion of IND-security for multiple messages. Then, we extend out notion of IND-security to protect against Chosen Plaintext Attack (CPA). We notice that CPA-security is at least as strong and more natural than multiple message security.

1 Minimal Assumption For Indistinguishable Encryption

Up to now, we have introduced a few mathematical entities in our theoretical framework, and a significant part of our work has been in realize their connections. For the sake of clarity, in fig. 1 we sketch the cryptographic primitives we have encountered so far, and how they are related to each other: an arrow from a primitive to another, means that the existence of the first implies the existence of the second.

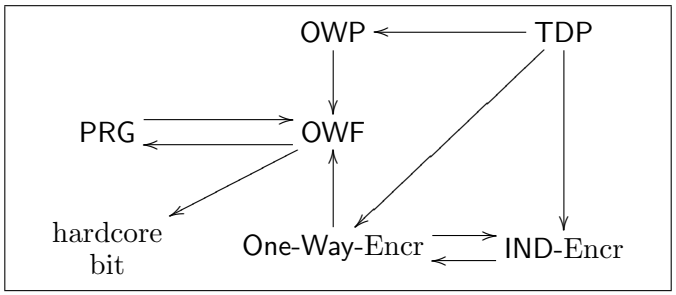

Figure 1: Relations between Cryptographic Primitives

It is worth noticing that the arrow from One-Way-Encryption to IND-Encryption does not mean that each PKE which is One-Way secure is also PKE secure (which is not the

case!). Instead, it means that the existence of a One-Way secure PKE implies the existence of an IND PKE as proved in the following theorem.

Theorem 1
If there exists a One-Way *secure* PKE $\mathcal{E} = (G, E, D)$ *then there exists an* IND PKE $\mathcal{E}' = (G, E', D')$.

Proof: The proof is an application of the One-Time Pad Lemma. By the hypothesis, we have that $\mathcal{E}$ is a One-Way secure PKE, i.e. the encryption function E is a OWF. Using the Goldreich-Levin theorem, we know that a random parity of the input bits of x is a hardcore predicate of E:

$$(E(x), r, (x \cdot r \bmod 2)) \approx (E(x), r, b), \quad \text{where } x, r \leftarrow_r M_k, b \leftarrow_r \{0,1\} \qquad (\dagger)$$

Let us define a one-bit PKE $\mathcal{E}' = (G, E', D')$, where to encrypt one bit b' using the randomness (x, r), E' runs E as follows:

$$E'(b'; x, r) = (E(x), r, ((x \cdot r) \bmod 2) \oplus b')$$

The key-generation algorithm for $\mathcal{E}'$ is exactly the same as for $\mathcal{E}$, while the decryption algorithm D' is the straightforward inverse of E':

$$D'(y, r, c) = (((D(y) \cdot r) \bmod 2) \oplus c).$$

To prove that $\mathcal{E}'$ is an IND PKE, we apply the One-Time Pad Lemma, choosing the distributions $X = (E(x), r)$ and $Y = (x \cdot r) \bmod 2$. Therefore, the $(\dagger)$ can be rewritten as $(X, Y) \approx (X, b)$. Thus, by the One-Time Pad Lemma, it holds that $(X, Y \oplus 0) \approx (X, Y \oplus 1)$, that is:

$$E'(0; x, r) = (E(x), r, ((x \cdot r) \bmod 2) \oplus 0) \approx (E(x), r, ((x \cdot r) \bmod 2) \oplus 1) = E'(1; x, r).$$

□

2 Efficient Encryption

So far, we have dealt with general constructions and properties of PKEs. However, those constructions, although polynomial, are still inefficient and generally not used in practice. Instead, people typically use optimized schemes based on *specific* number-theoretic assumptions (as opposed to general assumption like the existence of TDPs). In this section we study one of the sysmplest such PKEs, called the *ElGamal* cryptosystem, based on the hardness of the *Diffie-Hellman* Problem, which, in turn, is the basis of a famouse *Diffie-Hellman Key Exchange*.

2.1 The Diffie-Hellman Key Exchange

In their seminal paper on Public-Key Cryptography, W. Diffie and M. Hellman proposed the first *Key-Exchange Scheme*, i.e. a scheme to enable two parties, Alice and Bob, to exchange a jointly-selected shared key s to be used in a subsequent, secure communication (for example, as the shared key in a One-Time Pad cryptosystem). The scheme can be sketched as follows:

- Alice and Bob choose a public prime number p, and a generator g of the cyclic group $\mathbb{Z}_p^*$ (where $|p| = k$);
- Alice chooses (and keeps secret) an exponent $a \in \mathbb{Z}_{p-1}$, then computes $\alpha = g^a \bmod p$, and finally sends α to Bob;
- Bob chooses (and keeps secret) an exponent $b \in \mathbb{Z}_{p-1}$, then computes $\beta = g^b \bmod p$, and finally sends β to Alice;
- as soon as Bob receives the message α from Alice, he computes $s_B = \alpha^b \bmod p$;
- as soon as Alice receives the message β from Bob, she computes $s_A = \beta^a \bmod p$.

Clearly, at the end of this protocol, Alice and Bob share the secret key $s = g^{ab} \bmod p$:

$$s_A \equiv \beta^a \equiv (g^b)^a \equiv (g^a)^b \equiv \alpha^b \equiv s_B \pmod{p}$$

Notice that, while Alice or Bob can easily compute s (since they knows a and b respectively), an eavesdropper Eve has to face the much more difficult problem of computing g^{ab} given knowledge $(p, g, g^a \bmod p, g^b \bmod p)$: we will refer to this problem as the *Diffie-Hellman Problem* (DHP).

$$\begin{array}{rl} \text{INPUT:} & p,\ g \text{ generator of } \mathbb{Z}_p^*,\ (g^a \bmod p) \text{ and } (g^b \bmod p) \\ \text{OUTPUT:} & g^{ab} \bmod p \end{array}$$

But how can Eve succeed in her task? Of course, if Eve were able to compute the discrete logarithms in $\mathbb{Z}_p^*$, then she could compute $g^{ab} \bmod p$, by first extracting a from $g^a \bmod p$, and then raising $g^b \bmod p$ to the a^{th} power. It follows that Diffie-Hellman problem is at most as difficult as the more general Discrete Logarithm problem. Of course, this does not imply the equivalence of the two problems: it could certainly be the case that computing discrete logarithms is hard, while solving the Diffie-Hellman problem is possible. This leads to the following assumption:

DEFINITION 1 [Computational Diffie-Hellman (CDH) Assumption over $\mathbb{Z}_p^*$]
$\forall$ PPT algorithm A

$$\Pr[A(p, g, g^a, g^b) = g^{ab} \mid p\text{-prime}, |p| = k, a, b \leftarrow \mathbb{Z}_{p-1}, g \text{ generator of } \mathbb{Z}_p^*] = \mathsf{negl}(k)$$

$\diamondsuit$

In other words, the CDH assumption states that "it is hard to *completely* compute $g^{ab} \bmod p$". But this is not enough for key exchange, for the same reason that one-wayness is not enough to guarantee a secure encryption. Indeed, for all we know, the attacker may obtain a lot of partial information about the key g^{ab}, which would mean, for example, that it is not safe to use $s = g^{ab}$ as a one-time pad. What we need is that *the agreed key should look indistinguishable from random to Eve.* This leads to the following general definition.

DEFINITION 2 [Passive Security of Key Exchange, semi-formal] We say that a key exchange protocol is *secure against a passive*[1] *observer* if for any PPT attacker A we have $\langle T, s \rangle \approx$

[1] The reason for the term "passive" will become clear very soon.

$\langle T, R\rangle$, where T is the public transcript of the protocol, s is the key agreed upon by Alice and Bob, and R is a completely independent random key of the same length as s. $\diamondsuit$

For example, for the DH key exchange the transcript, which is the information that Eve has, consists of $T = \langle p, g, g^a, g^b\rangle$. Thus, in order for the DH key exchange to be passively secure it is necessary and sufficient to make the following stronger assumption, which is called *Decisional Diffie-Hellman (*DDH*) assumption*:

DEFINITION 3 [Decisional Diffie-Hellman (DDH) Assumption in $\mathbb{Z}_p^*$]
$\forall$ PPT algorithm A

$$\Big|\Pr[A(p,g,g^a,g^b,g^{ab}) = 1 \mid p\text{-prime}, |p| = k, a, b \leftarrow \mathbb{Z}_{p-1}, g \text{ generator of } \mathbb{Z}_p^*] -$$
$$\Pr[A(p,g,g^a,g^b,g^c) = 1 \mid p\text{-prime}, |p| = k, a, b, c \leftarrow \mathbb{Z}_{p-1}, g \text{ generator of } \mathbb{Z}_p^*\Big| =$$
$$= \mathsf{negl}(k)$$

Equivalently, it is infeasible to distinguish between g^{ab} and g^c given (p, g, g^a, g^b) (we omit p below):

$$(g, g^a, g^b, g^{ab}) \approx (g, g^a, g^b, g^c), \quad \text{where } a, b, c \leftarrow_r \mathbb{Z}_{p-1}$$

$\diamondsuit$

Lemma 1 *Under the* DDH *assumption in $\mathbb{Z}_p^*$, the DH key exchange is passively secure.*

Unfortunately, as stated, the DDH assumption is false!

Lemma 2 DDH *assumption is false in $\mathbb{Z}_p^*$.*

Proof: The problem is that Eve can learn the quadratic character of $g^{ab} \bmod p$. Recall, the quadratic character $\chi(x)$ for an element $x \in \mathbb{Z}_p^*$ is defined as: $\chi(x) = \begin{cases} 0 & \text{if } x \in QR_p \\ 1 & \text{if } a \notin QR_p \end{cases}$, where QR_p is the subgroup of quadratic residues modulo p. Equivalently, it is equal to the least significant bit (LSB) of the discrete logarithm of x base g. Either way, it can be efficiently computed by testing wther or not $g^{(p-1)/2} \bmod p \equiv 1$. Thus, from g^a Eve can compute $\mathsf{LSB}(a)$, from g^b — $\mathsf{LSB}(b)$, and then since $(p-1)$ is even

$$\mathsf{LSB}(ab \bmod (p-1)) = (ab \bmod (p-1)) \bmod 2 = ab \bmod 2 = \mathsf{LSB}(a) \oplus \mathsf{LSB}(b)$$

Put differently, g^{ab} is a quadratic character iff either both g^a abd g^b are, or none are. But this means that given the public transcript, Eve can compute the quadratic character $\chi(s)$ of the agreed key s, which means that s is distinguishable from ranndom (since a random x has a random quadratic character which cannot be predicted). In other words, the scheme leaks some information about the key. $\square$

It is common practice to fix this problem via an *ad hoc* solution, that consists in imposing a specific structure on the prime p. Let us assume that $p = 2q+1$, with p and q both primes (such a prime p is called *strong prime*). Then consider the subgroup G of $\mathbb{Z}_p^*$ made up by

all the quadratic residues modulo p: in other words $G \doteq QR_p$. It is known that the order of this subgroup is $\frac{p-1}{2} = q$, i.e. QR_p has prime order for our choice of p. Moreover, if h is a generator of $\mathbb{Z}_p^*$, then $g = h^2 \bmod p$ is a generator of G. Finally, $(G, \cdot)$ is isomorphic to $(\mathbb{Z}_q, +)$, since $g^a \cdot g^b \equiv g^{(a+b) \bmod q} \pmod p$. Indeed, by definition of *generator* and of *order* of a group, it holds that $g^q \equiv 1 \pmod p$. Now, writing $a + b = c \cdot q + (a + b) \bmod q$ for some c, we can make the following considerations:

$$g^a \cdot g^b \equiv g^{(a+b)} \equiv g^{c \cdot q} \cdot g^{(a+b) \bmod q} \equiv \left(g^q\right)^c \cdot g^{(a+b) \bmod q} \equiv 1^c \cdot g^{(a+b) \bmod q} \pmod p$$

Finally:

$$g^a \cdot g^b \equiv g^{(a+b) \bmod q} \pmod p .$$

We can now modify the Diffie-Hellman Key-Exchange protocol, so that the random values a and b chosen by Alice and Bob are drawn from $\mathbb{Z}_q$ (instead of $\mathbb{Z}_{p-1}$), and the whole computation is performed in G (instead of $\mathbb{Z}_p^*$).

The reason why these changes are effective is that now it is *always the case* that $g^a \bmod p$, $g^b \bmod p$ and $g^{ab} \bmod p$ are quadratic residues, and so what Eve can learn trough the considerations above (namely, the quadratic character of $g^{ab} \bmod p$) is already publicly know.

Of course, this "fix" does not a-priori *guarantee* that no other leakage of partial information is present by some other means. Nevertheless, extensive attempts to find other attacks on this modied scheme have failed, and, therefore, people came up with the following variants of the CDH and DDH assumptions in this new, slightly modified setting. Both variants are believed to be secure for a large enough security paprameter.

Definition 4 [Computational Diffie-Hellman (CDH) Assumption in QR_p]
$\forall$ PPT algorithm A

$$\Pr[A(g^a, g^b) = g^{ab} \mid p = 2q + 1, |p| = k, p, q \text{ primes}, a, b \leftarrow_r \mathbb{Z}_q, g \text{ generator of } G] = \mathsf{negl}(k)$$

(where it is omitted that A also knows the public modulo p and the generator g.) $\diamondsuit$

Definition 5 [Decisional Diffie-Hellman (DDH) Assumption in QR_p]
$\forall$ PPT algorithm A

$$\Big| \Pr[A(g^a, g^b, g^{ab}) = 1 \mid p = 2q + 1, |p| = k, p, q \text{ primes}, a, b \leftarrow_r \mathbb{Z}_q, g \text{ generator of } G] -$$
$$\Pr[A(g^a, g^b, g^c) = 1 \mid p = 2q + 1, |p| = k, p, q \text{ primes}, a, b, c \leftarrow_r \mathbb{Z}_q, g \text{ generator of } G] \Big| =$$
$$= \mathsf{negl}(k)$$

(where, again, it is omitted that A also knows the public modulo p and the generator g.) $\diamondsuit$

Similarly to Lemma 1, we get

Lemma 3 *Under the* DDH *assumption in $G = QR_p$, the modified DH key exchange is passively secure.*

We still have two more problems:

KEY IS NOT A "BIT-STRING". As it is defined, the DH key exchange outputs a key $s = g^{ab} \bmod p$ which looks like a random element of the group $G = QR_p$. As such, it appears that it cannot be directly used for a one-time pad or other general purposes. In other words, ideally we want to output a usual bit-string as our key. There are several ways out of this. One very general way is to usilize some "magic function" H from G to bit-strings of some length ℓ (roughly, $\ell \approx k$) with the property that a random element of G gets mapped to an almost random element of $\{0,1\}^\ell$. Turns out such general functions exist and called *randomness extractors*.

Here we describe a simple construction of such H for our specific choice of $G = QR_p$. Actually, let us first map from G to $\mathbb{Z}_q$ Since $p = 2q + 1$ and q is prime, q is odd. Hence, $q = 2k + 1$, and substituting we obtain $p = 4k + 3$, or alternatively $p \equiv 3 \pmod 4$. Then we know that any $s \in \mathbb{Z}_p^* = \{1 \dots 2q\}$, precisely only one among s and $-s$ is a quadratic residue. Similarly, precisely one of s and $-s$ is between 1 and q. Thus, we have a bijection between QR_p and $\mathbb{Z}_q$: map any $s \in QR_p$ either to s, if $s \le q$, or to $p - s$ is $s > q$.

Still, we only get a radnom number t from 1 to q, and q is not a precise power of 2. However, to get a bit-string which is very close to uniform we can simply truncate a few bits of t. Concretely, if ℓ is an integer s.t. $2^\ell < q$, then the "variation distance" between a random ℓ-bit string and our "extracted" string is less than $2^\ell/q$, so by dropping roughly 80 bits we get a nearly perfect random string.

ACTIVE ATTACKER. The second problem is that there is no way for Bob to know who is the sender of a message. In particular, the protocol we have allows Eve to mount the so called *person-in-the-middle* attack:

- Alice chooses (and keeps secret) an exponent $a \in \mathbb{Z}_q$, then computes $g^a \bmod p$, and sends it to Bob;
- Eve intercepts $g^a \bmod p$ (thus preventing Bob from getting it), and sends $g^{a'} \bmod p$ to Bob, for a random a' of her choice;
- Bob chooses (and keeps secret) an exponent $b \in \mathbb{Z}_q$, then computes $g^b \bmod p$, and sends it to Alice;
- Eve intercepts $g^b \bmod p$ (thus preventing Alice from getting it), and sends $g^{b'} \bmod p$ to Alice, for a random b' of her choice;
- when Bob receives the message $g^{a'} \bmod p$, he (erroneously) assumes that it comes from Alice, and thus he sets $s_B = (g^{a'})^b \bmod p = g^{a'b} \bmod p$;
- when Alice receives the message $g^{b'} \bmod p$, she (erroneously) assumes that it comes from Bob, and thus she sets $s_A = (g^{b'})^a \bmod p = g^{ab'} \bmod p$;
- Eve can easily compute both $s_A = (g^a)^{b'} \bmod p = g^{ab'} \bmod p$ and $s_B = (g^b)^{a'} \bmod p = g^{a'b} \bmod p$.

As a consequence of the attack, at the end of this run of the protocol, Alice and Bob happily enjoy their "secure" connection, exchanging their love messages and saying nasty

things about Eve, but $\ldots s_A \neq s_B$, and so actually they are talking trough Eve, who can not only learn everything, but also manipulate the communication. Notice also that the attack is generic: *any* protocol where Alice and Bob have no way to "authenticate" each other allows Eve to have 2 disjoint conversations with them just like below. Thus, to avoid this attack, a key point is to add *Authentication* to the basic Key-Exchange Scheme, i.e. provide some mechanisms that enables the receipient of a message to be sure about the identity of the sender. This is typically achieved by means of *Digital Signature*, which will be discussed later. When we do it, we will be able to get key exchange protocols resisting an *active* rather than passive attacker. But, for now, we need to assume that Eve is passive and merely listens to the conversation when trying to deduce information about the key s.

2.2 Back to Encryption: The ElGamal Cryptosystem

Now, we describe the ElGamal Cryptosystem, which exploits the hardness of the DHP to achieve a reasonable level of security quite efficiently.

1. On input 1^k, the *key-generation algorithm* Gen chooses a strong prime $p = 2q + 1$ along with a generator h of $\mathbb{Z}_p^*$, and sets $g = h^2 \bmod p$. Afterwards, G takes a random element $x \in \mathbb{Z}_q$ and sets $y = g^x \bmod p$. Finally, G outputs (PK, SK, M_k), where $PK = (p, g, y)$, $SK = x$, and the message space is $M_k = G = QR_p$.

2. Given a message $m \in M_k$ and some randomness r, the *encryption algorithm* E_{PK} outputs the ciphertext $(s, t) = (g^r, y^r \cdot m)$, where $PK = (p, g, y)$.[2] Notice that, since $y = g^x$, the ciphertext is actually of the form $(g^r, g^{xr} \cdot m)$, although, of course, this form is "hidden', and Eve can only see (s, t).

3. In order to decrypt a ciphertext (s, t), given the private key $SK = x$, the decryption algorithm D first recovers the quantity $s^x = g^{rx} = y^r$, and then 'simplifies' it out from t, computing $t \cdot (s^x)^{-1} = t \cdot (y^r)^{-1} = y^r \cdot m \cdot (y^r)^{-1} = m$.

4. In attacking the cryptosystem, the adversary Eve knows the public key $PK = (p, g, y = g^x)$ and a ciphertext $(s, t) = (g^r, y^r \cdot m)$, corresponding to m: overall, Eve's knowledge can be represented as $(g^r, g^x, g^{xr} \cdot m)$.

Comments.

- Both the encryption and the decryption algorithms of this scheme are fairly efficient, since, in each invocation, at most two modular exponentiations are required.

- Each ciphertext produced by the encryption algorithm is exactly twice as long as the plaintext. This space overhead is induced by the presence of some randomness in the encryption algorithm — which is unavoidable in any PKE that 'aims' to fulfil a notion of security that is stronger than One-Way security: as we noticed in previous lectures, a share of non-determinism is necessary to be able to send more than once the same message, in such a way that Eve cannot recognize that two different ciphertexts are related to each other.

[2] Where omitted, the computation is assumed modulo p.

- As it is defined, the ElGamal Scheme allows us to encrypt only messages that are elements of the group G. Usually, the messages we want to encrypt are not numbers, although they can easily be mapped onto numbers in some specific range (say, from 1 to q): but how can we force such numbers to be quadratic residues? We simply do the inverse mapping to the one described in the previous section, when we mapped a quadratic residue in $G = QR_p$ to an element of $\mathbb{Z}_q$. Namely, for our choice of p we showed that for any $m \in \mathbb{Z}_q$, precisely one of m and $-m$ belongs to G, so we simply apply the ElGamal encryption to either m or $-m$, whichever is the quadratic residue.

  Of course, we can turn this around, and directly encrypt $m \in \mathbb{Z}$, by changing the encryption to $(g^r, H(y^r)+m \bmod q)$, where H is the mapping described in the previous section: $H(z) = z$ if $z < q$ and $H(z) = -z$ if $z \geq q$. We will see, however, that this way we lose the homomorphic properties of ElGamal encryption that we describe next.

Homomorphic Properties. ElGamal Encryption has several very nice properties which we describe here informally.

- ElGamal encryption is *homomorphic*. Given the public key and two encryptions (s_0, t_0) and (s_1, t_1) of some *unknown* messages m_0 and m_1, one can efficiently compute an encryption $(s_0 s_1 \bmod p, t_0 t_1 \bmod p)$ of $m_0 m_1 \bmod p$. Indeed, if r_0 and r_1 are the coins used in the above encryptions, we have

$$(s_0 s_1 \bmod p, t_0 t_1 \bmod p) = (g^{r_0+r_1}, y^{r_0+r_1} \cdot (m_0 m_1)) = E(m_0 m_1;\ r_1 + r_2)$$

  More generally, for any integers i and j one can compute encryption of $m_0^i m_1^j \bmod p$ in a similar manner, by rasing computing $(s_0^i s_1^j, t_0^i t_1^j)$.

- ElGamal encryption is *blindable*. Given the public key, some encryption (s, t) of some *unknown* message m, and any value $m' \in G$, one can efficiently compute an encryption of $mm' \bmod p$. Indeed, this follows from the momomorphic property above, or can be seen directly by computing $(s, t \cdot m')$. Moreover, a *random* encryption of $m \cdot m'$ can be obtained by picking a random $r' \in \mathbb{Z}_q$ and computing $(s \cdot g^{r'}, t \cdot y^{r'} \cdot m')$. The verification of this fact is similar to the above.

- ElGamal encryption is *re-randomizable*. Given the public key and an encryption (s, t) of some *unknown* message m, one can compute a fresh *random* encryption of the same m which is identical to the process of really encrypting *known* m from scratch. Simply apply the previous scheme to $m' = 1$, or, directly, pick random r' and compute $(s \cdot g^{r'}, t \cdot y^{r'})$.

These homomorphic properties, by themself, are neither good nor bad: it depends upon the scenario at hand. Consider, for example, the case of a centralized authority C that is able to decrypt messages for others, but should not be allowed to learn the content of those messages. At first glance, this may seem hopeless, but the "blindable" property comes in help: instead of submitting the actual ciphertext, one can choose a random message m' and send to C a "blinded" ciphertext for $m \cdot m'$ (without knowng m, of course!). When C decrypts, it cannot understand the content (since the randomness of m' makes mm'

random!), but the other party can recover m by simply dividing out by m' (which it chose and knows).

On the other hand, in the setting of an auction, the "blindable" property would allow a participant to "cheat" by doubling the price offered by another participant, even when the auction bets are being encrypted with the ElGamal PKE of the auctioneer.

Relation to Key Encapsulation and Key Exchange. First, we notice that the ElGamal encryption also follows the general key encapsulation paradigm we studied earlier, except the XOR operation is replaced by multiplication in G. In particular, we get the following key encapsulation mechanism (Gen, E, D).

- Gen is exactly the same as for the ElGamal cryptosystem. It chooses a strong prime $p = 2q+1$ along with a generator h of $\mathbb{Z}_p^*$, and sets $g = h^2 \bmod p$. Afterwards, G takes a random element $x \in \mathbb{Z}_q$ and sets $y = g^x \bmod p$. Finally, G outputs (PK, SK, M_k), where $PK = (p, g, y)$, $SK = x$, and the message space is $M_k = G = QR_p$.
- The key encapsulation algortihm $KE(PK)$. It chooses a random r and sets key $s = y^r \bmod p$ and ciphertext $\psi = g^r \bmod p$.
- The key decapsulation algortihm $KD(\psi; SK)$ outputs $s = \psi^x \bmod p$.

The correctness is tested exactly as in the the DH key exchange: $\psi^x = g^{rx} = y^r$. And this is more than a coincidence. Indeed,

Observation 2 *Any two-round key exchange secure against a passive attacker yeilds a key encapsulation mechanism which is* CPA*-secure, and vice versa.*

Proof: First, we do the easier direction. Given any KEM, construct two-round key exchange as follows.

- In the first round, Alice choses $(PK, SK) \leftarrow \mathsf{Gen}(1^k)$ and sends PK to Bob.
- In the second round, Bob computes $(s, \psi) \leftarrow KE(PK)$, and sends ciphertext ψ to Alice. Bobn's key is s.
- Alice recovers $s = KD_{SK}(\psi)$.

The converse is also very similar, just view the first message as the public key PK of the KEM, the second message from Bob — as the ciphertext, Alice's coins for the first message as SK, and Bob's key as the value s. □

Applied to the DH key exchange, we *exactly get the above* KEM! Indeed, the public key is g^x, and encryption of the key g^{rx} is indeed g^r. Combining with Lemma 3, we get

Lemma 4 *Under the* DDH *assumption in QR_p, the ElGamal* KEM *is* CPA*-secure.*

Security of the ElGamal Cryptosystem. We can now use the following generalization of the one-time pad lemma to deduce a similar security for ElGamal encryption:

Lemma 5 (Generalized One-Time Pad Lemma) *Let $(G, \cdot)$ be a group, and R denote the uniform distribution over G. For all the distributions X, Y (not necessarily independent), if $(X, Y) \approx (X, R)$, then for all $m_0, m_1 \in G$, we have $(X, Y \cdot m_0) \approx (X, Y \cdot m_1)$. More generally, the "CPA-analog" of this result is true as well, where the PPT attacker is allowed to choose m_0, m_1 based on X.*

Proof: Just substitute $\cdot$ for $\oplus$ in the proof of the One-Time Pad Lemma (see Lecture 6). We leave the "CPA-analog" as an exercise. □

Using this result, we immediately get

Theorem 3 *Under the DDH assumption, the ElGamal cryptosystem is an IND secure PKE.*

Finally, we remark that we can also prove a weaker one-time secuirty of ElGamal under a weaker CDH assumption.

Theorem 4 *Under the CDH assumption, the ElGamal cryptosystem (or KEM) is a One-Way secure PKE.*

Proof: For the sake of contradiction, let us assume that the ElGamal cryptosystem is not a One-Way secure PKE:
$\exists$ PPT algorithm A such that:

$$\Pr[A(g^x, g^r, g^{xr} \cdot m) = m \mid p = 2q + 1, |p| = k, p, q \text{ primes}, x \leftarrow_r \mathbb{Z}_q, g \text{ generator of } G] = \epsilon$$

for some non-negligible ϵ.

Using a reduction approach, we now construct an algorithm A' which, given black-box access to the algorithm A, can efficiently solve the DHP with non-negligible probability, contradicting the hypothesis. On input (g^x, g^r), for random x and r in $\mathbb{Z}_q$, A' chooses $c \leftarrow_r \mathbb{Z}_q$ and runs $A(g^x, g^r, g^c)$: with probability ϵ, A will decrypt the 'ciphertext' g^c and thus will output $\tilde{m} = g^c \cdot (g^{xr})^{-1}$. At this point, A' computes $g^c \cdot \tilde{m}^{-1}$ as its guess for g^{xr}. Therefore, with probability roughly ϵ, A' will successfully solve the Diffie-Hellman Problem. □

Remark 1 *Quite interestingly, based on two different assumptions, it is possible to show that the* same *construction fulfil two different notions of security. Moreover, the assumption made and the security obtained are well-coupled: assuming that it is difficult to completely solve the DHP leads to the difficulty of completely breaking the ElGamal PKE; assuming that it is infeasible to learn anything useful about the solution of the DHP leads to the difficulty of learning anything useful about messages encrypted with the ElGamal PKE.*

2.3 Other Efficient Encryption Schemes

There are other IND-CPA-secure encryption schemes. In the homework we will study the original *Goldwasser-Micali (GM) cryptosystem*, which allows one to encrypt messages bit-by-bit (so it's not very efficient) and is secure under the quadratic residuosity assumption. The GM scheme is also homomorphic over $\mathbb{Z}_2$. A more recent efficient cryptosystem is called

a *Paillier* encryption, and is based on a stronger decisional variant of the RSA assumption which we will not state here. This schemes directly allows one to encrypt relatively large messages, but taken over group $\mathbb{Z}_n$, where n is a certain RSA-type modulus. Similarly to ElGamal, it is also homomorphic. Unlike ElGamal, it is homomorphic under modular *addition* rather than multiplication, so it's more convenient to use in many applications.

Finally, we will come back to public-key encryption when studying a stronger attack called chosen *ciphertext* attack (CCA), as opposed to the CPA attack we studied so far.

3 Secret-Key Encryption

We now return back to symmetric-key encryption. First, we remind ourselves the definition of Secret-Key encryption. (Notice the similarity to the definition of Public-Key encryption given in Lecture 6.)

DEFINITION 6 A Secret-Key encryption (SKE) is a triple of PPT algorithms $\mathcal{E} = (G, E, D)$, where

1. $G(1^k)$ as before outputs (PK, SK, M_k), PK, SK, M_K being a public key, secret key, and message space respectively. k is the security parameter.
2. $E(m; PK, SK, r)$ is the encryption algorithm that outputs ciphertext c. Note the difference: the input includes the secret key SK. The presence of the secret key contrasts SKE and PKE.
3. $D(c; SK)$, which is usually deterministic, outputs a decrypted message $\tilde{m} \in \{\mathsf{invalid}\} \cup M_k$.

As before, we require the correctness property: $\forall m \in M_k, \tilde{m} = m$. That is, if m was correctly encrypted using appropriate secret and public keys. ♢

For simplicity, we will omit PK and incorporate all PK information into SK. The use of PK may yield more efficient implementations because it may help reduce the size of the secret storage. Omitting the public key, however, will not cause any loss of generality in our discussion below. Thus, we often denote $SK = s$ and write $c \leftarrow E_s(m)$, $m \leftarrow D_s(m)$.

Now that SKE is defined, we will attempt to formally define the security notions for SKE.

3.1 One-Message IND-Security Against PK-Only Attack

We start with the simplest definition of security. Our first objective is to obtain security for one message only with adversaries unaware of any non-public information and unable to choose plaintext.

DEFINITION 7 SKE is one-message IND-secure against PK-only attack iff for any two messages, their ciphertexts are polynomially indistinguishable. $\forall m_0, m_1 \in M_k, E_s(m_0) \approx E_s(m_1)$. ♢

Example 1. One-Time Pad satisfies the above definition of security. Formally, $G(1^k) \rightarrow (\perp, R, M_k)$, where $M_k = \{0,1\}^k$, $R \in M_k$ is a random string.
Let $E_R(m) = m \oplus R$ and $D_R(c) = c \oplus R$. It is easy to see that $E_R(m_0) \equiv E_R(m_1)$, since both define truly random strings.

Example 2. One can reduce the size of the secret key in Example 1 by generating a pseudo-random string of size k using a PRG $G : \{0,1\}^k \rightarrow \{0,1\}^n$ where $n = p(k)$. Because $G(x) \approx \{0,1\}^{p(x)}$, we can use the one-time pad lemma to conclude that this more efficient version is one-message IND-secure.

The two examples above are no longer secure if more than one messages are to be encrypted. Indeed, $m_0 \oplus m_1 = c_0 \oplus c_1$ can be obtained by an adversary should he intercept ciphertexts c_0 and c_1 for m_0 and m_1. Therefore, we are in need of a better definition of IND-security for SKE. This is in contrast to the PKE, where one message security implied multiple message security.

3.2 SKE IND-Security With Respect To Multiple Messages

DEFINITION 8 SKE is IND-secure with respect to multiple messages against PK-only attack iff $\forall t = \mathsf{poly}(k), \forall m_0^1, m_0^2, ..., m_0^t$ and $m_1^1, m_1^2, ..., m_1^t \in M_k$,

$$E_s(m_0^1) \circ E_s(m_0^2) \circ \ldots \circ E_s(m_0^t) \approx E_s(m_1^1) \circ E_s(m_1^2) \circ \ldots \circ E_s(m_1^t)$$

$\diamondsuit$

In particular, this definition suggests that no information can be inferred about any of the m_i's as long as the number of transmitted messages is polynomial in k, the security parameter. Applying the hybrid argument, it can be shown that the following definition is equivalent to Definition 8.

DEFINITION 9 SKE is IND-secure with respect to multiple messages against PK-only attack iff $\forall t = \mathsf{poly}(k), \forall i \leq t, \forall m^1, m^2, \ldots, m^{i-1}, m_0^i, m_1^i, m^{i+1}, \ldots, m^t \in M_k$,

$$\begin{aligned} &E_s(m^1) \circ \ldots \circ E_s(m^{i-1}) \circ \quad E_s(m_0^i) \quad \circ E_s(m^{i+1}) \ldots \circ E_s(m^t) \approx \\ &E_s(m^1) \circ \ldots \circ E_s(m^{i-1}) \circ \quad E_s(m_1^i) \quad \circ E_s(m^{i+1}) \ldots \circ E_s(m^t) \end{aligned}$$

$\diamondsuit$

Notice, the last definition can be viewed the following way. The adversary chooses messages $m^1, m^2, \ldots, m^{i-1}, m^{i+1}, \ldots, m^t$, as well as a pair of messages m_0^i, m_1^i. Then the adversary gets to see the encryptions of the messages that it chose, and the encryption of m_b^i for a random bit b. The adversary then has to guess the bit b. Notice, since the adversary is unable to compute any of the encryptions by itself (unlike was possible in the PKE), it essentially means that A has *oracle access to the encryption oracle* $E_s(\cdot)$! Except this oracle access is "non-adaptive": all the messages have to be chosen at the beginning, and all the encryptions have to be given. From this point of view, it seems more natural to not place this unnatural restriction, and give A complete oracle access to $E_s(\cdot)$. Namely, at any point during its run, A can ask the oracle to encrypt any message m, and will get back

$E_s(m)$. At some point A chooses two messages m_0 and m_1. Then one of them m_b, will be encrypted, and $\tilde{c} \leftarrow E_s(m_b)$ will be given to A (for random unknown b). At this point A can again ask the oracle to encrypt a bunch of messages. Finally, A tries to predict b correctly. This is exactly the *chosen plaintext attack* (CPA). It also explains why we used the same terminology on the public-key setting: the explicit encryption oracle was implicitly given to the adversary by means of the public key PK!

We address IND-security against CPA in the next section.

3.3 SKE IND-Security Against Chosen Plaintext Attack (CPA)

The following definition summarizes what is said above. We denote by A^{E_s} the adversary who is given oracle access to $E_s(\cdot)$, as explained above.

DEFINITION 10 SKE is IND-secure against CPA iff $\forall \mathsf{PPT} B = (B_1, B_2)$,

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

$\diamondsuit$

To reiterate, this means that even if the adversary has an encryption oracle who will encrypt any message the adversary wants (without revealing the secret key), and if the adversary is free to produce any two messages and encrypt them using the oracle, and use any information thus obtained, including the ciphertexts of the two chosen messages, the adversary will still have a negligible advantage in guessing which of the two messages was encrypted by the experimenter.

Now we indicate that Definition 10 is at least as strong as Definition 8 (in fact, can be shown to be strictly stronger), i.e. a SKE that is IND-secure againts CPA can be safely used to transmit multiple messages.

Lemma 6 *If* SKE E *is* IND-CPA*-secure, then* SKE $\mathcal{E}'$ *with encryption function* $E'_s(m_0, \ldots, m_t) = E_s(m_0) \circ \ldots \circ E_s(m_t)$*, is also* IND-CPA*-secure, for any* $t = poly(k)$*.*

This result is proven using the hybrid argument in a completely identical manner than a similar result for the PKE was shown. Indeed, having oracle access to the encryption function allows the adversary to prepare encryptions of $m_1^0, \ldots, m_{i-1}^0, m_{i+1}^1, \ldots, m_t^1$ which are needed for the hybrid argument. The inability to make such encryptions is exactly the reason the proof fails in general for one-message IND-secure SKE's.