1 Public Key Encryption (cont’d)

As lecture notes, the reader is encouraged to review the notes on Structuring Security Proofs as Sequences of Games and IND-CCA2 Security.

Remember that we gave the definition of IND-CPA security model against public key encryption in lecture 7. In this lecture, we will extend the security models to be resilient against much stronger adversaries.

1.1 Games as a General Proof Methodology

In this lecture we will formalize the proofs of security as a sequences of games. Referring the definition of Games below from the IND-CCA2 notes, we only covered Section 3 and Section 4 of IND-CCA2 notes. You are only required to read these relevant parts.

Definition 1 (Games). Let \( G \) be deterministic program that takes input \( \lambda \) bits and operates in \( v \) steps employing the random variables \( \rho_1, \ldots, \rho_v \) that are sampled uniformly from the domains \( R_1, \ldots, R_v \) (e.g., \( R_\ell = \{0, 1\}^{s_\ell(\lambda)} \) where \( s_\ell : \mathbb{N} \to \mathbb{N} \)). \( G \) operates in \( v \) steps where every step is using randomness \( \rho_\ell \) sampled uniformly from a space that is parameterized on the input of the game as well as it may depend on previous variables. The \( \ell \)-th step is of the form:

\[
y := f(y_1, \ldots, y_d, \rho_1, \ldots, \rho_\ell); \quad \rho_\ell \xleftarrow{} \mathcal{R}_{y_1, \ldots, y_d}^{y_1, \ldots, y_d}
\]

where \( y \) is a variable name, \( y_1, \ldots, y_d \) are all variables defined in the previous steps, \( f \) is a deterministic polynomial-time computable function, \( \rho_1, \ldots, \rho_\ell \) are the random coin tosses of the steps 1, \ldots, \( \ell \), and each \( \rho_\ell \) is selected uniformly at random from a space \( R_\ell^{y_1, \ldots, y_d} \).

In addition to steps as defined above the program \( G \) may contain other deterministic instructions such as conditional statements (if-then-else) and for-loops that may cause a set of steps to be repeated (in such case each repetition is assumed to be a different step). In all cases the whole program flow can be unfolded to a straight-line or tree-structure (based on conditionals) sequence of assignments and function calls. We assume that the program returns a single bit as outcome.

Let \( T \) be the event that the program outputs 1 (the probability space is comprised by all variables \( \langle \rho_1, \ldots, \rho_v \rangle \)). Such \( G \) will be called a “game” and the event \( T \) will be called the winning event.

Example. Let PRIME be a probabilistic procedure that uses coin tosses in Coins, takes as input \( 1^\lambda \), and returns as output a \( \lambda \) bit prime.

<table>
<thead>
<tr>
<th>Game ( G_1 ) on input ( 1^\lambda )</th>
<th>Random variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( p \leftarrow \text{PRIME}(1^\lambda, \rho) )</td>
<td>( \rho \xleftarrow{} \text{Coins} )</td>
</tr>
<tr>
<td>2. ( x \leftarrow \sum_{i=0}^{\lambda-1} 2^i \rho_i )</td>
<td>( \rho_1, \ldots, \rho_\lambda \xleftarrow{} {0, 1} )</td>
</tr>
<tr>
<td>3. Output the least significant bit of ( x \mod p )</td>
<td></td>
</tr>
</tbody>
</table>
Note that the following methodology will be best understood if read in parallel to the example proofs of the Notes. We will now simply describe the general structure that you may observe how the security proof is achieved in the Notes:

1. Start from the attack game where the adversary is being used as a black box.
2. Modify the game to another game so that the probability distribution of two games is related and we know the difference.
3. Repeat (2) until it is self-evident that we know the output distribution of the game.

2 Beyond IND-CPA Security; Lunch-Time Attacks and Malleability

Although IND-CPA security is well-defined and models the chosen plaintext attacks, it doesn’t support security against chosen ciphertext attacks or malleability. To be more precise, an encryption algorithm is said to be malleable if it is possible for an adversary to transform a ciphertext into another ciphertext which decrypts to a related plaintext. For instance, we proved in Lecture 7 that ElGamal cryptosystem is IND-CPA secure, but it is still malleable because of the following:

Given a ciphertext \( \langle A, B \rangle = \langle g^r, h^r M \rangle \), it is possible to compute the encryption of \( M^w \), by simply transforming \( \langle A, B \rangle \) to \( \langle g^r A^w, h^r B^w \rangle \).

Towards modelling security against malleability, we will try to formalize the power that the adversary has. Our first attempt will be granting the adversary to learn the decryption of messages corresponding to a list of non-adaptively chosen ciphertexts. In a non-adaptive chosen ciphertext attack, known as lunch-time attacks, the adversary has access to the decryption machine for a time-period to analyze the decryptions results later. This attack scenario in an extreme case may allow the adversary to recover the scheme’s secret decryption key by issuing carefully chosen ciphertexts.

In Section 4 of the Notes, we define the security against non-adaptive chosen ciphertext (IND-CCA1 model of security) attacks and a variant of the two-generator ElGamal Public-key Cryptosystem that is IND-CCA1 secure. For the readers who are interested in security (IND-CCA2 model) against adaptively-chosen ciphertext attacks we refer to the Section 5. Note also that the IND-CCA2 security is equivalent to the non-malleability property.