CS 7810 Graduate Cryptography September 27, 2017

Lecture 7: CPA Security, MACs, OWFs

Lecturer: Daniel Wichs *Scribe: Eysa Lee*

1 Topic Covered

- Chosen Plaintext Attack (CPA)
- MACs
- One Way Functions (OWFs)
- NP puzzles
- Impagliazzo's worlds

2 Chosen Plaintext Attack (CPA)

Recall last lecture we introduced a new notion of security, chosen plaintext attack (CPA). In this model, the adversary is allowed a polynomial number of plaintext queries to an oracle, and the oracle replies with the encryption of these messages. In a challenge phase, the adversary sends two messages m_0, m_1 and receives an encryption of one of the messages. The adversary is allowed polynomial many more oracle encryption queries, but is finally challenged to identify which of the two messages was encrypted.

We can formally define CPA security using $\mathsf{CPA\text{-}Game}^b$, where $b \in \{0,1\}$. $\mathsf{CPA\text{-}Game}^b$ is defined as follows:

1. The challenger chooses a key $k \leftarrow \{0,1\}^n$
2. The adversary repeatedly chooses messages m_i and the challenger sends back ciphertexts $c_i = \mathsf{Enc}(k, m_i)$. The adversary can do this as many times as it wants.
3. The adversary chooses two challenge messages m_0^*, m_1^*. The challenger sends back a ciphertext $c^* = \mathsf{Enc}(k, m_b^*)$.
4. The adversary repeatedly chooses messages m_i and the challenger sends back ciphertexts $c_i = \mathsf{Enc}(k, m_i)$. The adversary can do this as many times as it wants.
5. The adversary outputs guess $b' \in \{0,1\}$.

Definition 1 An encryption scheme is CPA secure if $\mathsf{CPA\text{-}Game}^0 \approx \mathsf{CPA\text{-}Game}^1$. In other words, for all PPT adversaries A,

$$\left| \Pr[\mathsf{CPA\text{-}Game}_A^1(n) = 1] - \Pr[\mathsf{CPA\text{-}Game}_A^0(n) = 1] \right| = \mathsf{negl}(n)$$

$\diamondsuit$

2.1 Encryption Scheme with CPA Security

We note that it is not possible for a deterministic encryption scheme to be CPA secure. This is because we do not have any restrictions on which messages the adversary can query encryptions of. That is, the adversary is allowed to query the same message m multiple times or even oracle query the challenge messages m_0, m_1. For a deterministic encryption scheme, it is trivial to see that an adversary can easily compare the ciphertext produced from $\mathsf{Enc}(k, m_b)$ against oracle encryption queries of m_0 and m_1.

Let $F_k : \{0,1\}^n \to \{0,1\}^l$ be a PRF. We can use F_k to construct a randomized encryption scheme. The construction works as follows:

$\mathsf{Enc}(k, m)$: $x \leftarrow \{0,1\}^n$. Output ciphertext $\mathsf{c} = (x, F_k(x) \oplus m)$.

$\mathsf{Dec}(k, \mathsf{c})$: Let c be of the form (x, ct). Output decryption $\mathsf{ct} \oplus F_k(x)$.

This encryption scheme resembles one time pad, except that we use a PRF F and a randomly sampled input to the PRF x to hide the message m. Intuitively this scheme works because, despite being given x in the ciphertext, F_k looks like random and $F_k(x)$ cannot be computed without knowledge of k by the security of the PRF scheme.

Theorem 1 *This encryption scheme is CPA secure. That is, with the above encryption scheme, $\mathsf{CPA\text{-}Game}^0 \approx \mathsf{CPA\text{-}Game}^1$.*

Proof: We construct a hybrid argument to prove security of the scheme. We begin and end with $\mathsf{CPA\text{-}Game}^0$ and $\mathsf{CPA\text{-}Game}^1$ and define intermediate hybrids that are computationally indistinguishable. The hybrids are as follows.

H^0: $\mathsf{CPA\text{-}Game}^0$

H^1: Let R be a random function. We define this experiment to be the same as $\mathsf{CPA\text{-}Game}^0$ except that we replace all computations of $\mathsf{Enc}(k, m)$ (in steps 2,3,4 of the game) with a procedure that samples $x \leftarrow \{0,1\}^n$ and outputs $(x, R(x) \oplus m)$.

H^2: We define this experiment to be the same as H^1 except that we change how the ciphertext c^* in step 3 of the game is computed. In particular, we just choose $c^* = (x^*, y^*) \leftarrow \{0,1\}^n \times \{0,1\}^l$ uniformly at random.

H^3: Let R be a random function and $x \leftarrow \{0,1\}^n$. We define this experiment to be the same as $\mathsf{CPA\text{-}Game}^1$ except that we replace all computations of $\mathsf{Enc}(k, m)$ (in steps 2,3,4 of the game) with a procedure that samples $x \leftarrow \{0,1\}^n$ and outputs $(x, R(x) \oplus m)$.

H^4: $\mathsf{CPA\text{-}Game}^1$

First, we will show $H^0 \approx H^1$ by constructing a reduction between the two. Suppose A could distinguish between H^0, H^1. Notice, the only difference between H^0, H^1 is that the PRF F in H^0 is replaced by a random function R in H^1. We can construct a reduction algorithm B to distinguish a PRF from random. A plays $\mathsf{CPA\text{-}Game}^0$ with B acting as an encryption oracle to A. On a message m_i, B samples a random $x_i \leftarrow \{0,1\}^n$ and sends x_i to the an oracle O. The oracle O returns $O(x_i)$, which is either the evaluation of x_i on a PRF

F or on a random function R. The reduction algorithm B then sends to A the encryption as $(x_i, O(x_i) \oplus m)$. We can see when O uses a PRF F, A is in hybrid H^0. When O uses a random function R, A is in hybrid H^1. It is clear that if such an A existed that could distinguish between H^0 and H^1, then it would be possible for B to break the security of F. If we assume F is secure, then it must be $H^0 \approx H^1$.

Next, we show $H^1 \approx H^2$. We note that if y^* is uniformly random then the distributions of $y^* \oplus m$ is identical to the distribution of y^*. Furthermore, $y^* = R(x^*)$ is uniformly random as along as x^* is "fresh" meaning that it is different from all of the values x_i chosen by the encryption procedure during stages 2 and 4 of the game. Therefore, if x^* is fresh then the games H^1, H^2 are identical and the statistical distance between the two games is just the probability that x^* is not fresh. Furhtermore, since x^* is chosen unfiromly at random, the probability that it is not fresh is at most $q(n)/2^n$ where $q(n)$ is the total number of encryption queries that the adversary makes in stages 2,4 of the game. Since the adversary is polynomially bounded, $q(n)$ is a polynomial and therefore $q(n)/2^n$ is negligible. This shows that the two games are statistically close and even a computationally unbounded adversary (that only makes polynomially many encryption queries) cannot distinguish them. Therefore, $H^1 \approx H^2$.

Now, we observe the hybrids are symmetric about H^2. That is, the pairs H^0, H^1 and H^4, H^3 are identical apart from the message m_b encrypted in the challenge phase. Therefore, the same proof for $H^0 \approx H^1$ can be applied to H^3 and H^4. The same proof for $H^1 \approx H^2$ can be applied to prove $H^2 \approx H^3$.

We have $H^0 \approx H^1 \approx H^2 \approx H^3 \approx H^4$, and therefore $H^0 \approx H^1$. This means $\mathsf{CPA\text{-}Game}^0 \approx \mathsf{CPA\text{-}Game}^1$. □

3 MACs

Let us revisit MACs, which were introduced in the first lecture. We now consider them against computationally bounded adversaries. Like in the CPA game, an adversary is allowed a polynomial number of queries, which it will receive the MAC of the queried message. We define Mac-Game as follows.

1. The challenger chooses a key $k \leftarrow \{0,1\}^n$
2. The adversary sends to the challenger a query message m_i. The challenger sends back the authentication tag $\sigma_i = \mathsf{MAC}(k, m_i)$. The adversary is allowed to repeat this as many times as it wants.
3. Finally, the adversary sends a pair m^*, σ^* and the game outputs 1 if and only if $\mathsf{MAC}(k, m^*) = \sigma^*$ and m^* had not been previously queried by the adversary (i.e., $m^* \notin \{m_i\}$).

Definition 2 A MAC scheme is computationally secure if for all PPT adversaries A we have

$$Pr[\mathsf{Mac\text{-}Game}_A(n) = 1] = \mathsf{negl}(n)$$

♢

Theorem 2 *Let F be a PRF $\{0,1\}^l \to \{0,1\}^n$. $\mathsf{MAC}(k,m) = F_k(m)$ is a computationally secure MAC scheme.*

Proof: First, we will show $\mathsf{Mac\text{-}Game} \approx \mathsf{R\text{-}Game}$, where in $\mathsf{R\text{-}Game}$ we replace F_k with a random function R (both in stage 2 where we create authentication tags for the adversary as well as in stage 3 where we verify the adversary's tag). Suppose there exists some adversary A that could distinguish $\mathsf{Mac\text{-}Game}$ and $\mathsf{R\text{-}Game}$. Since $\mathsf{R\text{-}Game}$ only replaces F_k with R, it is clear that A could be used to distinguish F_k from R, breaking the security of PRF. Therefore, we have that $\mathsf{Mac\text{-}Game} \approx \mathsf{R\text{-}Game}$.

Next, we claim $\Pr[\mathsf{R\text{-}Game}_A(1) = 1] \leq 2^{-n}$. The MAC in $\mathsf{R\text{-}Game}$ uses a random function R, so the MAC of any message m_i is $R(m_i)$. We note the adversary must produce a pair m^*, σ^* and wins if $R(m^*) = \sigma^*$. Since the adversary can not have queried m^* previously and R is completely random, the probability that the adversary guesses correctly and $R(m^*) = \sigma^*$ is 2^{-n}.

Combining the two, we have that

$$\begin{aligned}\Pr[\mathsf{Mac\text{-}Game}_A(1) = 1] &\leq \Pr[\mathsf{R\text{-}Game}_A(1) = n] + \mathsf{negl}(n)\\ &\leq 2^{-n} + \mathsf{negl}(n)\\ &\leq \mathsf{negl}(n)\end{aligned}$$

We have showed $\Pr[\mathsf{Mac\text{-}Game}_A(n) = 1] = \mathsf{negl}(n)$, so the MAC scheme is computationally secure. □

4 One Way Functions (OWF) and NP puzzles

We now introduce one way functions.

Definition 3 A function $f : \{0,1\}^* \to \{0,1\}^*$ is one way if:

- f can be computed in polynomial time
- $\forall\mathsf{PPT}$ adversaries A :

$$\Pr[f(x') = y : x \leftarrow \{0,1\}^n, y = f(x), x' \leftarrow A(1^n, y)] = \mathsf{negl}(n)$$

♢

Intuitively, if f is one way, f should be easy to compute in th forward direction (given x it's easy to find $f(x)$), but it should be hard to invert (given $y = f(x)$ for a random x it should be hard to find any preimage of y). We note f does not necessarily need to be one-to-one. There may be many preimages of y and it should be hard to find *any* of them.

Definition 4 An NP puzzle consists of a poly-time computable relation $R(y,x)$. We think of y as a puzzle or statement, and x as a solution or witness. The relation R tests if x is a good solution for the problem y and outputs 1 (accept) or 0 (reject). ♢

Definition 5 An average hard NP puzzle has a PPT algorithm $\mathsf{Gen}(1^n)$ that generates hard puzzles that have a solution. In particular, for $y \leftarrow \mathsf{Gen}(1^n)$, then $\exists x$ where $R(y,x) = 1$. Furthermore $\forall\mathsf{PPT}$ adversaries A

$$\Pr[R(y,x') = 1 : y \leftarrow \mathsf{Gen}(1^n), x' \leftarrow A(1^n, y)] = \mathsf{negl}(n)$$

$\diamondsuit$

DEFINITION 6 A one way NP puzzle has a PPT algorithm $\mathsf{Gen}(1^n)$ that generates puzzles together with a solution: for $(y, x) \leftarrow \mathsf{Gen}(1^n)$ we have $R(y, x) = 1$. Furthermore $\forall\mathsf{PPT}$ adversaries A

$$\Pr[R(y, x') = 1 : (y, x) \leftarrow \mathsf{Gen}(1^n), x' \leftarrow A(1^n, y)] = \mathsf{negl}(n)$$

$\diamondsuit$

Theorem 3 *One-way puzzles exist if and only if One-way functions exist.*

Proof: Given a OWF it's easy to create a one-way puzzle consisting of a relation R which is defined by $R(y, x) = 1$ if $f(x) = y$ and a PPT algorithm $\mathsf{Gen}(1^n)$ which samples a random $x \leftarrow \{0, 1\}^n$ and outputs $(f(x), x)$.

The other direction is more challenging. Assume the $\mathsf{Gen}(1^n)$ algorithm uses n-bits of randomness, and denote $(y, x) \leftarrow \mathsf{Gen}(1^n; r)$ to be a run of the algorithm with input r. We can define the function $f(r)$ which runs $(y, x) \leftarrow Gen(1^n; r)$ and outputs y. Any adversary A that inverts the OWF can also be used to break the one-way puzzle. Given a puzzle y we can call $A(y)$ which outputs r'. If A inverts the OWF f it means that $\mathsf{Gen}(1^n; r') = (y, x')$. By correctness $R(y, x') = 1$ and therefore x' is a good solution to the puzzle y. If Gen uses more then n bits of randomness, say some polynomial n^c for some constant $c > 0$, then we can "rescale" it and define $\mathsf{Gen}'(1^n; r) = \mathsf{Gen}(1^{n^{1/c}}; r)$ to be a one-way relation that uses only n-bits of randomness.

$\square$

5 Impagliazzo's Worlds

We note if $P = NP$, then for any NP puzzle, we can find a solution x given problem y (if a solution exists). However, what if $P \neq NP$? We can imagine different worlds or scenarios and consider their impliciation for computer science and cryptography. In particular, we will look at five worlds imagined by Russell Impagliazzo:

- Algorithmica: In this world, $P = NP$ or $BPP = NP$. This means we can solve all NP puzzles in the worst case or there exists a randomized algorithm that can solve NP puzzles in the worst case.
- Heuristica: $P \neq NP$ in this world, but average hard NP puzzles don't exist. In other words, for every efficiently samplable distributions on puzzles, there is some heuristic algorithm that solves the puzzles generated from that distribution, even if there is no universal algorithm that always solves all puzzles.
- Pessiland: This is the world where average-hard NP puzzles exist but one-way puzzles don't exist. This world is considered the worst of all worlds. There are average hard puzzles that we cannot heuristically solve but we also cannot meaningfully uses them to build crypto applications.

- Minicrypt: This world where one-way NP puzzles and therefore also one-way functions exist. In other words, it is possible to generate hard puzzles with a solutions. In this world we can do most of symmetric key cryptography such as PRGs, PRFs, MACs and CPA encryption (and few other things as we will see). But it might still be the case that public-key encryption doesn't exist in this world.
- Cryptomania: This is the world where we can do public-key encryption and many other advanced cryptographic applications that come with it.