Lecture 18 - Chosen Ciphertext Security

Boaz Barak

November 21, 2005

Public key encryption We now go back to public key encryption. As we saw in the case of private key encryption, CPA security is *not* sufficient for many applications of encryption. For example, consider the login problem:

- Client and server share secret PIN.
- Client and server share secret crypto key.
- To login, client sends encryption of PIN to server.

We saw that even if the encryption is CPA secure, an adversary that controls the communication channel might be able to find out the PIN. However, we were able to solve this problem using a MAC.

Public key login One problem with this protocol is that the client and the server need to share a cryptographic key. This is often not realistic (we can trust a user to memorize a PIN, but nothing more than that). A much more realistic setting is that the client only knows a *public key* of the server. This means that we have the following protocol:

- Client and server share secret PIN.
- Client and server has public encryption key e of server.
- To login, client sends encryption of PIN with key e to server.

However, this protocol is not going to work if the encryption scheme is only CPA secure. (See exercise) Even worse, we can't fix this with signatures in the way we fixed the previous protocol with MACs, since don't want to assume that the client has a private signature key with the server knowing the corresponding public key.

This means that we need a stronger notion of encryption. Indeed, for this and many other applications, we need encryption schemes that are *chosen ciphertext secure* (CCA).

Chosen Ciphertext Security (CCA) (This is a straightforward conversion of the private-key definition of lecture 9 to the public key setting)

Definition 1 (CCA security). An encryption $(\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ is said to be (T, ϵ)-*CCA secure* if it's valid (for every $(e, d) = \mathsf{Gen}(1^n)$, $\mathsf{D}_d(\mathsf{E}_e(x)) = x$) and for every T-time A if we consider the following game:

- $(e, d) \leftarrow_{\mathrm{R}} G(1^n)$.
- A gets as input e.
- A gets access to black boxes for $\mathsf{E}_e(\cdot)$ (redundant) and $\mathsf{D}_d(\cdot)$.
- A chooses x_1, x_2.
- Sender chooses $i \leftarrow_{\mathrm{R}} \{1, 2\}$ and gives A $y = \mathsf{E}_e(x_i)$.
- A gets more access to black boxes for $\mathsf{E}_e(\cdot)$ (redundant) and $\mathsf{D}_d(\cdot)$ but is restricted not to ask y to the decryption box. More formally, A gets access to the following function $D'_d(\cdot)$ instead of $\mathsf{D}_d(\cdot)$

$$D'_d(y') = \begin{cases} D_d(y') & y' \neq y \\ \perp & y' = y \end{cases}$$

  ($\perp$ is a symbol that signifies "failure" or "invalid input")
- A outputs $j \in \{1, 2\}$.

A is successful if $j = i$, the scheme is (T, ϵ) *CCA-secure* if the probability that A is successful is at most $\frac{1}{2} + \epsilon$.

It's not hard to show that the hardcore based CPA-secure public key encryption scheme we saw in class is *not* CCA secure. In the homework exercises you will show that CCA security suffices to solve the login problem. CCA security is now considered the preferred notion of security for encryption schemes, and the one that corresponds best to "digital envelopes".

Plan Unfortunately, we'll probably *not* get to see in this course a construction of an encryption scheme with a proof that it is CCA secure under some standard computational assumption. Rather, we'll show an encryption scheme with a proof that it is CCA secure in the random oracle model.

A CPA secure scheme in the random oracle model . For starters, we'll show a *CPA* secure scheme in the random oracle model. One advantage of this scheme over the hardcore-bit based scheme we saw before will be that it will have shorter ciphertexts. (To encrypt n bits we'll need $3n$ bits as opposed to n^2 in the previous bit-by-bit scheme.) The main advantage of concern to us will be that we'll be able to generalize it to a CCA secure encryption scheme.

A CPA Secure Scheme:

- Let $G : \{0,1\}^n \to \{0,1\}^n$ be a random oracle and $\{(f, f^{-1})\}$ be collection of trapdoor permutations. The public key of the scheme will be $f(\cdot)$ while the private key be f^{-1}.
- To encrypt $x \in \{0,1\}^n$, choose $r \leftarrow_{\mathrm{R}} \{0,1\}^n$ and compute $f(r), G(r) \oplus x$.
- To decrypt y, z compute $r = f^{-1}(y)$ and let $x = r \oplus z$.

Theorem 1. *The above scheme is CPA secure in the random oracle model.*

Proof. For public key encryption, the encryption oracle is redundant and so CPA security means that an adversary A that gets as input the encryption key ($f(\cdot)$ in our case) cannot tell apart $\mathsf{E}(x^1)$ and $\mathsf{E}(x^2)$ for every x^1, x^2.

However, in the random oracle model we need to give A also access to the random oracle $G(\cdot)$.

We denote the ciphertext A gets as challenge by y^*, z^* where $y^* = f(r^*)$ and $z^* = G(r^*) \oplus x^*$. We start by proving the following:

Claim 1.1. *The probability that A queries r^* of its oracle $G(\cdot)$ is negligible.*

Proof. Consider the following experiment: instead of giving $z^* = G(r^*) \oplus x^*$, we give A the string $z^* = u \oplus x^*$ where u is a uniform element. The only way A could tell apart the two cases is if he queries r^* to G and sees that the answer is different from u, but then we already "lost". Thus, the probability that A queries r^* in this experiment is the same as the probability that it queries r^* in the actual attack.

However, in this experiment the only information A gets about r^* is $f(r^*)$ - thus if it queries $G(\cdot)$ the value r^* then it inverted the trapdoor permutation! $\square$

Now this means we can ignore the probability that A queried r^* and hence we can (like in the proof of the claim) assume that $z^* = u \oplus x^*$ where u is chosen independently at random. However, this means that

A gets *no information* about x^* and hence will not be able to guess if it's equal to x^1 or x^2 with probability greater than 1/2.

□

The CCA secure encryption First note that if we have one random oracle we can have many independent oracles (just have $G_i(x) = G(i \circ x)$). We'll use two independent random oracles G, H in the next scheme.

- Let $G, H : \{0,1\}^n \to \{0,1\}^n$ be two independent random oracles and $\{(f, f^{-1})\}$ be collection of trapdoor permutations. The public key of the scheme will be $f(\cdot)$ while the private key be f^{-1}.
- To encrypt $x \in \{0,1\}^n$, choose $r \leftarrow_{\text{R}} \{0,1\}^n$ and compute $f(r), G(r) \oplus x, H(x, r)$.
- To decrypt y, z, w compute $r = f^{-1}(y)$ and let $x = r \oplus z$. Then, check that $w = H(x, r)$: if so then return x, otherwise return $\perp$.

Theorem 2. *The above scheme is CCA secure.*

Proof. Let A be an algorithm in a CCA attack against the scheme. Again, denote by y^*, z^*, w^* the challenge ciphertext A gets where $y^* = f(r^*)$, $z^* = G(r^*) \oplus x^*$ and $w^* = H(x^*, r^*)$.

Since H is a random oracle, we can assume that throughout the attack, no one (the sender, receiver or A) will ever find a two pairs x, r and x', r' such that $x \circ r \neq x' \circ r'$ but $H(x, r) = H(x', r')$.

Thus, at each step i of the attack and for every string $w \in \{0,1\}^n$ we can define $H_i^{-1}(w)$ in the following way: if the oracle H was queried before with some x, r and returned w then $H_i^{-1}(w) = (x, r)$. Otherwise, $H_i^{-1}(w) = \perp$.

We also observe that a pair x, r completely determines a ciphertext y, z, w that is a function of x and r and also that y, z completely determine x and r.

We consider the following experiment: at step i, we answer a decryption query y, z, w of A in the following way: if $H_i^{-1}(w)$ is equal to some x, r that determine y, z, w then return x. Otherwise, return $\perp$.

Note that the difference between this oracle and the real decryption oracle is that we may answer $\perp$ when the real decryption oracle would give an actual answer. However, we claim that A will not be able to tell apart with non-negligible probability the difference between this decryption oracle and the real one. Indeed, the only difference would be if A managed to ask the oracle a query: y, z, w satisfying the following:

- $w \neq w^*$ (since if $w = w^*$ then we have that $H_i^{-1}(w) = x^*, r^*$ and hence A either asked a query that both oracles answer with $\perp$ or it asked the disallowed query y^*, z^*, w^*).
- w was not returned as the answer of any previous query x, r to $H(\cdot)$ by A.
- If we let x, r be the values determined by y, z then $H(x, r) = w$. However, since (x, r) was not asked before, the probability that this happens is only 2^{-n}.

Thus, we see that we can simulate the decryption box of A without knowing f^{-1}, x^* and r^*. This means that A basically has no use for the decryption box and hence it would be sufficient to prove that the scheme is just CPA secure. This proof follows in a similar way to the previous scheme. □

Some practical issues One drawback of this scheme is that it uses a ciphertext of length $3n$ where n is the length of input for the trapdoor permutation. Scheme that use n-bit long ciphertext are known in the literature (see web page for links to papers).