6.876/18.426: Advanced Cryptography April 2, 2003

Lecture 14: Lunchtime and Chosen Ciphertext Security

Scribed by: David Wilson

1 Recap: Naor/Young's Lunchtime Security

1.1 Definition

At the beginning of class we reviewed the concept of security against a "lunchtime" attack (abbreviated CCA-1). Recall the protocol-based definition of CCA-1 security:

1. *Initiation*: An adversary ADV_{PPT} is given the public key PK.
2. *Learning*: ADV can send polynomially many ciphertexts to an oracle, who will send the corresponding plaintexts (or will notify ADV if the ciphertext is invalid). In this way, ADV attempts to learn about the secret key or weaknesses in the cryptosystem.
3. *Testing*: ADV then sends two messages m_0, m_1 which he thinks he can distinguish. The decryptor chooses a random $b \in \{0,1\}$ and sends the encryption of m_b. ADV then responds with a guess $g \in \{0,1\}$.

Def. A cryptosystem is *CCA-1 secure* (or "lunchtime-secure") iff

$$\Pr(b = g) < \frac{1}{2} + \frac{1}{Poly(k)}$$

where k is the security parameter.

1.2 Naor-Yung Cryptosystem

The Naor-Yung scheme[NY90] is a cryptosystem $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ defined as follows:

G: The public key is the triplet (PK_1, PK_2, R), where PK_1 and PK_2 are two independent public keys generated from an underlying GM-secure cryptosystem $(\mathsf{G}, \mathsf{E}, \mathsf{D})$, and R is a random string long enough to use for a non-interactive zero-knowledge proof. The secret key is (SK_1, SK_2), the two secret keys associated with PK_1 and PK_2.

E: To encrypt a message, first encrypt it using each of the keys PK_1, PK_2 using the underlying GM-secure cryptosystem. Also, supply a non-interactive zero-knowledge proof (using R as the random string) that the two encryptions are of the same message. Thus, $\mathsf{E}(m) = (\mathsf{E}(m, PK_1), \mathsf{E}(m, PK_2), \Pi_R(\text{"same message"}))$.

D: The decryption algorithm has two steps:

   1. First, check the validity of Π. If Π is invalid, terminate with failure ($\perp$).

2. Assuming Π is valid, calculate $\mathsf{D}_1(\mathsf{E}_1(m)) = m$.

It is important to first check that Π is a valid proof. If Π is invalid, then it is entirely possible that the two encryptions are not of the same message. The importance of this point will become clear in the proof.

2 Proof of Lunchtime Security

Here we proved that the Naor-Yung system described above is CCA-1 secure. Proof, as always, is by contradiction.

2.1 Proof

Assume that there exists some ADV_{PPT} that can guess b with probability non-negligibly better than $\frac{1}{2}$. We show how to construct, given ADV, an algorithm adv that breaks the GM-security of the underlying cryptosystem $(\mathsf{G}, \mathsf{E}, \mathsf{D})$. Construct algorithm adv as follows:

1. adv randomly selects $a \in \{1, 2\}$. adv then generates (PK_a, SK_a) using G, and lets PK_{3-a} be the input key PK (that adv will break). adv must also generate a "random" string for the NIZK proofs. In the case of some NIZK proof systems, for example, that of Feige, Lapidot, and Shamir [FLS90], the simulator can actually pick the random string in advance, without knowing the theorem it will be asked to prove. We thus use the simulator to generate the string R, and send the Naor-Yung public key (PK_1, PK_2, R) to ADV. We will need to record the state α of the simulator so we can use it later.

   Note that R may not be random if generated in this way. However, we know that R generated this way must be indistinguishable from random, or the zero-knowledgeness of the NIZK proof system would not hold.

2. ADV sends a ciphertext (c_1, c_2, Π) to adv. Just as the usual case, adv first checks that Π is valid with respect to R; if invalid, $\perp$ is returned to ADV. If it is valid, then presumably, $m_1 = m_2 = m$, so adv can return m (obtained by decrypting c_a using SK_a). Since the messages are the same, ADV obtains no knowledge of the procedure used to decrypt the ciphertext, merely the decryption itself. *(This step can be repeated a polynomial number of times.)*

   Note here that if the adversary manages to slip a false proof in, we might have a problem. If c_1 and c_2 encrypt different messages, then the adversary can distinguish the case $a = 1$ (in which case we are behaving exactly as we're supposed to) from the case $a = 2$. We will need to argue later that the adversary cannot distinguish these two cases. However, this cannot happen with more than negligible probability by the soundness of the NIZK proof system.

3. ADV sends two messages (m_0, m_1) to adv. adv returns (m_0, m_1) and gets back $c = E(m_b, PK)$. The goal of adv is to guess b. We next pick a random bit r and let $c_a = E(m_r, PK_a)$ and $c_{3-a} = c$. Then, we run the simulator from state α to fake a

proof $\tilde{\Pi}$ that c_1 and c_2 encrypt the same message. We send $(c_1, c_2, \tilde{\Pi})$ to ADV. ADV returns a guess g, and adv outputs the same value.

Now we prove that the probability that adv returns the correct value is non-negligibly better than 1/2.

Case 1: $r = b$

If $r = b$, then from the hypothesis ADV will guess correctly with probability nonnegligibly better than $\frac{1}{2}$. For simplicity, we can state that in this case $g = b$ with probability 51%.

Case 2: $r \neq b$

If $r \neq b$, then ADV has received one encryption of each message and a "proof" that they are the same message. Although ADV's behavior is not well defined in this circumstance, the protocol constrains ADV to send a single bit. Since even the order in the public key is randomized (the reason for the use of a above), ADV cannot distinguish between the case $m_b = m_0$ and $m_b = m_1$ since ADV doesn't know which message encrypts m_b. Thus, ADV is correct with probability exactly 50%.

Since r is selected randomly, Case 1 and Case 2 occur with equal probability; thus, overall the probability that $g = b$ is 50.5%. However, this violates the GM-security of the underlying cryptosystem, since adv can then simply send g to the owner of the copied public key as a guess for b and be correct with probability nonnegligibly more than $\frac{1}{2}$.

Here we also see the importance of adv first checking Π before giving the plaintext m, since otherwise ADV could send encryptions of two different messages and thus determine that adv always returned the message m_a. Once this is known, the argument given in case 2 above is invalid, since ADV can use the position in the message to differentiate m_r from m_b. ADV could then return r with probability 51% (without violating its protocol, since its behavior is undefined for an invalid input), resulting in adv having no advantage in breaking the GM-secure public key and invalidating the proof of lunchtime security given here.

2.2 A Few Concerns

2.2.1 Can adv really fool ADV?

In Step 3, adv provides an NIZK proof that $m_r = m_b$, when this is not necessarily the case (and, even if true, is not known by adv). However, adv can construct a valid-looking $\tilde{\Pi}$ since adv previously prepared the random string R for exactly this purpose. This is exactly the method used by the simulator in Lectures 9 and 10 to provide a false proof.

Concerns were raised in class about R not being random. However, several lectures previously it was proven that a simulator could prove 3SAT $\in$ NIZK using a string that was indistinguishable from random; thus, adv can prove the NP statement "$m_r = m_b$" using a string that appears random to ADV.

2.2.2 Can ADV fool adv?

It is worth noting that although R was constructed in order for adv to construct a false proof, ADV uses the same public key to send ciphertexts to adv in Step 2. Thus, if ADV

could use R to construct a $\tilde{\Pi}$ that appears valid, he could potentially determine that *adv* always answers m_a and adjust his final guess accordingly, skewing the probability.

The resolution of this concern again rests on the fact that R is indistinguishable from random. *adv* can construct a false proof using R because *adv* knows a trapdoor function that gives an unusual probability distribution of results when applied to R. ADV only sees the string R, and cannot construct a false proof since he does not know the appropriate secret.

3 Adaptive Chosen Ciphertext Security

3.1 Definition

The definition of adaptive chosen ciphertext security (also called *CCA-2 security*) is very similar to that of CCA-1 security, except that the adversary is allowed an additional "question-and-answer" period after receiving the encrypted m_b. Thus, ADV's queries may depend on the encrypted message m_b, whereas in the previous case ADV had to make a guess immediately.

1. *Initiation*: An adversary ADV_{PPT} is given the public key PK.

2. *Learning Period 1*: ADV can send polynomially many ciphertexts to an oracle, who will send the corresponding plaintexts (or will notify ADV if the ciphertext is invalid). In this way, ADV attempts to learn about the secret key or weaknesses in the cryptosystem.

3. *Testing*: ADV then sends two messages m_0, m_1 which he thinks he can distinguish. The decryptor chooses a random $b \in \{0,1\}$ and sends the encryption of m_b.

4. *Learning Period 2*: ADV can again send polynomially many ciphertexts to the decryptor/oracle, who will again send the corresponding plaintexts (or will notify ADV if the ciphertext is invalid). Thus, ADV's queries in this step can depend on the encryption of m_b received in Step 3.

5. *Guess*: ADV sends a guess $g \in \{0,1\}$.

Def. A cryptosystem is *CCA-2 secure* (or "adaptive chosen ciphertext secure") iff

$$\text{Prob}(b = g) < \frac{1}{2} + \frac{1}{Poly(k)}$$

where k is the security parameter.

3.2 The Naor-Yung System and CCA-2 Security

Since we have just finished proving that the Naor-Yung cryptosystem is CCA-1 secure, the question naturally arises as to whether it is CCA-2 secure. The answer is, not necessarily. Recall that the security of the Naor-Yung system relied on ADV not being able to produce a false proof $\tilde{\Pi}$ based only on seeing R (and not knowing the appropriate trapdoor). However, once adv sends the encryption $(\mathsf{E}_1(m_r), \mathsf{E}_2(m_b), \tilde{\Pi})$, ADV has an example of a false proof $\tilde{\Pi}$ to work with as well. While ADV cannot come up with a false proof on his own, it is possible that seeing $\tilde{\Pi}$ will give him the knowledge necessary to create a second false proof $\tilde{\Pi}'$. Once ADV has the ability to create false proofs, he can potentially break the security of the cryptosystem, as shown in the previous section. Thus, the Naor-Yung system is not CCA-2 secure unless creating $\tilde{\Pi}'$ is hard.

The system would be secure if the underlying cryptosystem used in it was actually non-malleable, but non-malleability for a cryptosystem is equivalent to CCA-2 security, so that would be too much of an assumption.

4 A CCA-2 Secure Cryptosystem

The professor then gave the outline of a CCA-2 secure cryptosystem by Sahai [Sah99]. The general idea is recorded here; further elaboration can be found in the paper.

Sahai's system extends the basic notion of NIZK and uses the modified notion to provide CCA-2 security.

4.1 Intuition

Intuitively, there are several properties that are desirable for an NIZK proof system to make the Naor-Yung construction secure against CCA-2. Briefly:

A. If I see a good proof Π of a true theorem X, and use it to produce a good proof Π' of another theorem X', then not only is X' true, but I could have generated (X', Π') already.

B. If I see a good-looking proof $\tilde{\Pi}$ of a false theorem $\tilde{X}$, and use it to produce a good-looking proof Π' for another theorem X', then not only is X' true, but I could have generated (X', Π') already.

4.2 Construction

NIZK proofs have taken the basic form of $(\mathsf{P}, \mathsf{V}, \mathsf{S}, f)$, where P is the prover, V is the verifier, S is the simulator, and f is the length of a public random string σ required to complete the proof. Sahai's enhanced NIZK algorithm uses $(\bar{\mathsf{P}}, \bar{\mathsf{V}}, \bar{\mathsf{S}}, \bar{f})$ works as follows:

1. Consider a digital signature scheme (VK, SK) that is not existentially forgeable under a chosen message attack. Let $\bar{\sigma} = \sigma_1\sigma_2...\sigma_{2k-1}\sigma_{2k}$, where k is the length of the verification key VK. Thus, where in a regular NIZK proof σ is a random string of a certain length, $\bar{\sigma}$ is a random string of $2k$ times that length which can be viewed as the concatenation of $2k$ separate strings. Thus, $\bar{f}(k) = 2kf(k)$.

2. The prover $\bar{\mathsf{P}}$ carries out a non-interactive zero-knowledge proof of the theorem for each bit of the verification key VK, using an associated piece of $\bar{\sigma}$. For example, if the first bit of VK is 0, then $\bar{\mathsf{P}}$ generates Π_1 using σ_1; if it is 1, then $\bar{\mathsf{P}}$ instead generates Π_2 using σ_2. Similarly, for the i-th bit of PK, $\bar{\mathsf{P}}$ generates Π_{2i-1} using σ_{2i-1} if the bit

is 0 and Π_{2i} using σ_{2i} if the bit is 1. In this way, $\bar{\mathsf{P}}$ generates k separate NIZK proofs for the theorem. For the other k indices, we let $\Pi_i = \perp$.

3. Using the signing key SK, $\bar{\mathsf{P}}$ then signs the concatenation of the proofs generated in the previous step along with the theorem itself:

$$s = \mathrm{Sig}_{SK}(\Pi_1, ...\Pi_{2k}, X)$$

.

4. A proof $\bar{\Pi}$ then consists of s, $\Pi_1, \ldots, \Pi_{2k}$, and the verification key VK. Verifying the proof is straightforward: all Π_is must be valid proofs of the theorem if i is an index that would be selected in step 2 for an actual proof, the signature must be valid, and the Π_is must use the pieces of $\bar{\sigma}$ corresponding to the bits of VK. Note that the indices chosen are based on VK so the verifier can determine which are supposed to be real proofs and which may be omitted.[1]

4.3 Proof Sketch of CCA-2 Security

The general idea is as follows. Given ADV we construct adv as before, except after the challenge is given, we answer queries on ciphertexts other than the challenge we gave to ADV just as we would have before the challenge. Now, if ADV never generates any false proofs after the challenge, the argument from before works. However, there is a concern that the adversary might gain the capability to make a false proof after seeing the fake proof we give in the challenge.

Suppose that we give the adversary a proof $s, VK, \bar{\Pi}$ of a statement X, and the adversary produces a proof $s', VK', \bar{\Pi}'$ of a statement X' such that at least one thing is different: either $X \neq X', s \neq s', VK \neq VK'$, or $\bar{\Pi} \neq \bar{\Pi}'$. There are two cases.

Case 1: $VK' = VK$
If the same verification key is used, the adversary has successfully signed a new message using the same verification key. However, this contradicts the assumption that the digital signature scheme being used is unforgeable.[2]

Case 2: $VK' \neq VK$
If $VK' \neq VK$, the two must differ in at least one bit. Assume WLOG that i is such that $VK[i] = 0$ and $VK'[i] = 1$. Then in the construction of $\bar{\Pi}'$, the adversary necessarily constructed Π_{2i} using σ_{2i}. However, the adversary has not seen a proof using σ_{2i} before, since the original prover used σ_{2i-1} in his proof instead. Thus, the adversary must have already been able to construct an NIZK proof of X' before seeing the proof $\bar{\Pi}$, so by the soundness of the underlying NIZK proof system, if X' is false, the adversary can only produce such a proof with negligible probability.

[1] Actually, one detail we need is that the signature scheme used is one where for any message and any verification key, there is only ONE valid signature of that message under that key. However, this is not hard to construct. For details, see Sahai's paper.

[2] The case where $s \neq s'$ but all other components are the same is ruled out by the unique signature requirement from the footnote above.

4.4 Closing questions

Several things to think about with regard to the Sahai scheme:

- Does this scheme satisfy CCA-2 security? What, exactly, does it accomplish?
- The intention of CCA-2 security is to protect against an adversary who may be "inspired" by seeing a proof of a false theorem. What about security against an adversary who sees multiple false proofs?
- As it stands, the Sahai scheme works to prove a single theorem. Can you modify it to work with polynomially many theorems?

References

[FLS90] U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero-knowledge proofs based on a single random string. Proceedings, 31st IEEE Symposium on Foundations of Computer Science (FOCS), pp. 308–317, 1990.

[NY90] M. Naor, M. Yung. Public-Key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. Proceedings, ACM 22nd Annual Symposium on Theory of Computing (STOC), pp. 427–437, 1990.

[Sah99] A. Sahai. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. Proceedings, 40th IEEE Symposium on Foundation of Computer Science (FOCS), pp. 543–553, 1999.