6.876/18.426: Advanced Cryptography 31 March 2003

Lecture 13: NIZK and the Lunchtime Attack

Scribed by: Jonathan Herzog

1 Overview

In this lecture, we explored the first part of the first application of non-interactive zero-knowledge (NIZK). In particular, we explored how NIZK can be used to construct a public-key encryption scheme that is secure against the *chosen-ciphertext attack*. We will explain exactly what the chosen-ciphertext attack *is* in the next section, but first a comment about what we actually accomplished.

To complete this task, we would need to

1. Give a definition of the chosen-ciphertext attack and security against it,
2. Give a particular construction, and
3. Prove that the construction satisfies the definition.

In this lecture, we did the first two but not all of the third. Instead, we managed to prove that the construction satisfied a slightly weaker form of security known as the *lunchtime* attack.[1] This is because our scheme is actually *not* secure against the chosen-ciphertext attack; we need additional tools before that can be achieved. However, the final scheme is only a slight variation of the one given here.

2 The Chosen-Ciphertext attack

In this section, we define the chosen-ciphertext attack and security against it. This attack applies to public-key encryption:

Definition 1 *A public-key encryption scheme is a triple of algorithms* $(\mathsf{G}, \mathsf{E}, \mathsf{D})$*, where*

- $\mathsf{G} : \mathsf{Parameter} \times \mathsf{Coins} \to \mathsf{PublicKey} \times \mathsf{PrivateKey}$ *is the randomized key generation algorithm,*
- $\mathsf{E} : \mathsf{String} \times \mathsf{PublicKey} \times \mathsf{Coins} \to \mathsf{Ciphertext}$ *is the randomized encryption algorithm, and*
- $\mathsf{D} : \mathsf{Ciphertext} \times \mathsf{PublicKey} \times \mathsf{PrivateKey} \to \mathsf{String}$ *is the decryption algorithm.*

Very often we write $\mathsf{G}(1^k)$ (assuming $\mathsf{Parameter} = 1^*$) for the distribution induced by $\mathsf{G}(1^k, r)$ where r is chosen uniformly from Coins. We also write $\mathsf{E}(m, pk)$ for the distribution induced by $\mathsf{E}(m, pk, r)$ where r is chosen randomly. (We also insist that decryption with the right secret key undoes the encryption operation.)

[1] In the notation of [1], we showed the scheme to be CCA-1 secure, not CCA-2.

2.1 Static Adversaries

What does it mean for an encryption scheme to be secure? Against a static adversary, the gold standard for security is that of *semantic* or *general message (GM)* security. Informally, this definition states that no efficient (PPT) adversary has a non-negligible chance of telling whether an encryption contains message m_0 or message m_1 as plaintext, even if the adversary chooses m_0 and m_1 itself. More formally,

Definition 2 *A public-key encryption algorithm* $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ *is* semantically secure *if:*

$$\forall \mathsf{Adv}_{PPT^2} = (\mathsf{Adv}_1, \mathsf{Adv}_2),\ \forall \textit{ polynomials } q, \forall \textit{ sufficiently large } k,$$

$$\begin{aligned} \Pr[\ & (pk, sk) \leftarrow \mathsf{G}(1^k); \\ & m_0, m_1, s \leftarrow \mathsf{Adv}_1(1^k, pk); \\ & b \leftarrow \{0,1\}; \\ & c \leftarrow \mathsf{E}_{pk}(m_b); \\ & g \leftarrow \mathsf{Adv}_2(1^k, pk, m_0, m_1, c, s) \\ g = b\ &] \leq \tfrac{1}{2} + \tfrac{1}{q(k)} \end{aligned}$$

Note: This is not exactly the definition given in class. The definition given in class is the "three-pass" version of semantic security, meaning that m_0 and m_1 do not depend on the choice of public key pk. In this, more convenient version, the adversary can choose the messages after examining the public key. For further discussion, see [2].

2.2 Adaptive Adversaries

The adaptive adversary is very similar to the static adversary. It is certainly trying to play the same game: distinguishing an encryption of m_0 from an encryption of m_1. It still gets to choose m_0 and m_1 based on the public key pk. However, it now has access to some help: decryption oracles. Suppose that before the adversary chooses the messages m_0 and m_1 it can query a decryption oracle (decryption with the secret key sk, that is) as many times as it likes on whatever input it likes. Then the adversary is launching a *lunchtime attack*:

Definition 3 *A public-key encryption algorithm* $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ *is* secure against a lunchtime attack *if:*

$$\forall \mathsf{Adv} = (\mathsf{Adv}_1, \mathsf{Adv}_2), \forall \textit{ polynomials } q, \forall \textit{ sufficiently large } k,$$

$$\begin{aligned} \Pr[\ & (pk, sk) \leftarrow \mathsf{G}(1^k); \\ & m_0, m_1, s \leftarrow \mathsf{Adv}_1^{O(\cdot)}(1^k, pk); \\ & b \leftarrow \{0,1\}; \\ & c \leftarrow \mathsf{E}_{pk}(m_b) : \\ & \mathsf{Adv}_2(1^k, pk, m_0, m_1, c, s) = b\ ] \leq \tfrac{1}{2} + \tfrac{1}{q(k)} \end{aligned}$$

where $O(\cdot) = \mathsf{D}(sk, \cdot)$.

If the adversary has access to the decryption oracle not only before choosing m_0 and m_1 but also after receiving the challenge ciphertext c, then it can launch a fully adaptive chosen ciphertext attack. To keep the definition from being trivial, we insist that post-challenge oracle refuse to decrypt c. It will, however, decrypt any other input:

Definition 4 *A public-key encryption algorithm* $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ *is* secure against an adaptive chosen-ciphertext attack *if:*

$$\forall \mathsf{Adv} = (\mathsf{Adv}_1, \mathsf{Adv}_2), \forall \textit{ polynomials } q, \forall \textit{ sufficiently large } k,$$

$$\begin{aligned}\Pr[\quad & (pk, sk) \leftarrow \mathsf{G}(1^k); \\ & m_0, m_1, s \leftarrow \mathsf{Adv}_1^{O_1(\cdot)}(1^k, pk); \\ & b \leftarrow \{0,1\}; \\ & c \leftarrow \mathsf{E}_{pk}(m_b): \\ & \mathsf{Adv}_2^{O_2(\cdot)}(1^k, pk, m_0, m_1, c, s) = b \quad] \leq \tfrac{1}{2} + \tfrac{1}{q(k)}\end{aligned}$$

where $O_1(\cdot) = \mathsf{D}(sk, \cdot)$ *and* $O_2(\cdot) = \mathsf{D}(sk, \cdot)$ *if the input is not equal to* c *and* $\perp$ *if it is.*

It is these definitions that we will use in our proof.

3 The Construction

The following construction is due to Naor and Yung [3]. The basic intuition is that the sender should not only encrypt, but also provide a proof that he knows the plaintext. If this can be done, then the adversary gains no information during the chosen-ciphertext attack: it *already* knows the answer it would get from the decryption oracle.

Although we cannot do exactly this, we do something close:

Let $(\overline{\mathsf{G}}, \overline{\mathsf{E}}, \overline{\mathsf{D}})$ be a semantically secure encryption scheme. Let $(\mathsf{P}, \mathsf{V}, \mathsf{S})$ be a NIZK proof scheme for the language

$$\mathcal{L} = \left\{ e_1, e_2, pk | \exists m, r_1, r_2 \text{ such that } e_1 = \overline{\mathsf{E}}(m, pk, r_1) \wedge e_2 = \overline{\mathsf{E}}(m, pk, r_2) \right\}$$

This language is in $\mathcal{NP}$, and for any $(e_1, e_2, pk) \in \mathcal{L}$, a witness is the (m, r_1, r_2) from the language definition.

Then, let:

- G, on input 1^k, run $\overline{\mathsf{G}}$ on input 1^k **twice** to produce two key pairs (pk_1, sk_1) and (pk_2, sk_2). It also produces a "sufficiently long" random string R, and outputs $pk = (pk_1, pk_2, R)$ and $sk = (sk_1, sk_2)$.
- E, on input $(m, (pk_1, pk_2, R))$, produces random strings r_1 and r_2 by flipping coins. It then runs $\overline{\mathsf{E}}(m, pk_1, r_1)$ to produce e_1, runs $\overline{\mathsf{E}}(m, pk_2, r_2)$ to produce e_2, and runs P on e_1, e_2, m r_1 and r_2 (using R as the reference string) to produce a proof π. It then outputs (e_1, e_2, π).
- D, on input (e_1, e_2, π), (pk_1, pk_2, R) and (sk_1, sk_2), first verifies the proof π using V and the reference string R . If the proof verifies, it decrypts e_1 using $\overline{\mathsf{D}}$ and sk_1.

Question: Why is it important that the decryption oracle verify π? For that matter, why should we even use the proof system? Why not just decrypt e_1 and e_2, and test to see if they contain the same plaintext?

Answer: The answer is that we don't want the decryption algorithm to reveal any information to the adversary that the adversary would not already know in creating a ciphertext. Hence, unless the adversary *proves* that the ciphertexts contain the same plaintext,

the decryption oracle reveals that information to it. (And since the proof system guarantees soundness, the adversary must know the mutual plaintext of the two encryptions to produce a valid proof π.)

Question: What if there are multiple senders? Then we have multiple provers all using the same reference string R without any sort of synchronization. Will this be a problem?

Answer: No, it won't. We need a NIZK proof system that can handle multiple provers that don't necessarily synchronize — and they exist. (Alternately, we can provide a different reference string to each sender, but this is inelegant and breaks the public-key model.)

4 Proof of Security

4.1 Completeness

Is this a public-key encryption scheme? That is, if an honest sender encrypts m to produce e, will the honest receiver decrypt e to produce m? With overwhelming probability, the honestly-generated proof π will verify (due to the completeness of the proof system.) And with overwhelming probability, decrypting an honestly made e_1 with sk_1 will produce the original message m. So yes, this is a public-key encryption scheme.

Question: (From Professor) Can we get perfect completeness? That is, can we get a scheme were the honest verifier will get the plaintext with probability 1?

Answer: (From Chris) Yes, since NIZK is an AM(2), and we can always get a proof system for AM(2) with perfect completeness. (How? Take a proof system that has completeness with probability almost 1. If you find yourself in the unlikely situation where you would not get completeness with the intended system, simply send the witness or message instead.)

4.2 Soundness

Will the adversary ever be able to get a "decryption" of a *malformed* ciphertext (e_1, e_2, π)? No, because the underlying proof system is sound.

4.3 Security against the Lunchtime Attack

(The Professor titled this section ZK because the proof ultimately derives from the fact that the proof system is ZK. But security against the lunchtime attack it what is ultimately proven.)

Suppose that there exists an adversary $\mathsf{Adv} = (\mathsf{Adv}_1, \mathsf{Adv}_2)$ which can successfully complete a lunchtime attack against this scheme. That is, suppose

$$\exists \mathsf{Adv} = (\mathsf{Adv}_1, \mathsf{Adv}_2), \exists \text{ a polynomial } q, \text{ for infinitely many } k,$$

$$\Pr[\quad (pk, sk) \leftarrow \mathsf{G}(1^k);\\ m_0, m_1, s \leftarrow \mathsf{Adv}_1^{O(\cdot)}(1^k, pk);\\ b \leftarrow \{0,1\};\\ c \leftarrow \mathsf{E}_{pk}(m_b):\\ \mathsf{Adv}_2(1^k, pk, m_0, m_1, c, s) = b \quad] \geq \tfrac{1}{2} + \tfrac{1}{q(k)}$$

where $O(\cdot) = \mathsf{D}(sk, \cdot)$. Then we can use Adv to construct adv, an adversary that violates the semantic security of $(\overline{\mathsf{G}}, \overline{\mathsf{E}}, \overline{\mathsf{D}})$. Here is how we construct adv:

- adv_1, on input 1^k and pk (where pk is a public key of $(\overline{\mathsf{G}}, \overline{\mathsf{E}}, \overline{\mathsf{D}})$):
  1. Flips a coin to select $p \leftarrow \{1, 2\}$. Without loss of generality, let $p = 1$, meaning that the ciphertext adv will try to break will be in the *second* position. (This will be clearer in a second.)
  2. Runs $\overline{\mathsf{G}}(1^k)$ to produce pk_1 and sk_1.
  3. Uses the simulator S of the NIZK proof scheme to generate a reference string R.
  4. Gives (pk_1, pk, R) to Adv_1.
  5. When Adv_1 makes a request to the decryption oracle on ciphertext (e_1', e_2', π'), it verifies the proof and (if it passes) returns the output of $\overline{\mathsf{D}}(e_1', sk_1)$.
  6. When Adv_1 produces the messages m_0 and m_1 (with state information s, adv_1 outputs m_0, m_1, and $s' = (pk_1, R, s, s'')$, where s'' is whatever state information the simulator S needs to maintain.
- adv_2, on input 1^k, pk, m_0, m_1, c, $s' = (r, pk_1, R, s, s'')$:
  1. Creates the "ciphertext" for Adv_2 from c by picking $r \leftarrow \{0, 1\}$, encrypting m_r under public key pk_1 to create c', and running the simulator (with state information s'') to create a "proof" π.
  2. Runs Adv_2 on input 1^k, (pk_1, pk, R), m_0, m_1, (c', c, π) and s.
  3. When Adv_2 returns a guess g, adv_2 returns g.

The ciphertext c given to adv_2 will be the encryption of m_b where $b \in \{0, 1\}$. With what probability will the g returned by adv_2 equals b? Here we only sketch the proof, but present the essential idea.

There are two cases:

1. If $r = b$, then adv_2 chose to encrypt the same message as was given to it in c. Hence, the input given to Adv_2 is two encryptions of m_b and a "proof" of that fact. If the proof is indistinguishable from a real proof (which it must be, at least computationally, due to the definition of the simulator S) then this is exactly the input expected by Adv_2. Hence, Adv_2 (and thus adv_2) will return b with probability at least $\frac{1}{2} + \frac{1}{q(k)}$.
2. In the other case, adv_2 decides to encrypt the *other* message. Hence, Adv_2 receives as input one encryption of m_0 and one encryption of m_1. Recall that p, the position of the input ciphertext, was chosen randomly. Hence, the input contains exactly no information about which encryption was the input to adv_2 and which was created by it. Hence, Adv_2 has exactly a 50% chance of being able to return as g the message which was adv_2's input.

The two cases are equally likely. Thus, the chances that adv_2 returns the correct value as g is

$$\frac{1}{2}\frac{1}{2} + \frac{1}{2}\frac{1}{q(k)} = \frac{1}{2} + \frac{1}{2q(k)}$$

which is still non-negligible.

Question: As adv simulates Adv, it must choose the reference string R before it knows the theorem to be "proven". (That is, before it knows which encryptions must be proven to have the same plaintext.) Can this be done?

Answer: Yes, and this is why I (Professor) made such a big deal about it when I demonstrated it.

Question: (From Professor) Does this proof still work if we move to a fully adaptive chosen-ciphertext attack? That is, does it still work if Adv_2 has access to a decryption oracle as well?

Answer: Go home and think about it. [Scribe note: the answer is "no," but I'm not telling you why.]

5 Choice Quotes from The professor

- "When it's convenient to me, I lie whenever I can."
- "I was wise enough to not write down what I wanted to prove; that helps me."

References

[1] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology (CRYPTO 98)*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer–Verlag, 1998. Full version found at http://www.cs.ucsd.edu/users/mihir/papers/relations.html.

[2] Silvio Micali, Charles Rackoff, and Bob Sloan. The notion of security for probabilistic cryptosystems. *SIAM Journal on Computing*, 17(2):412–426, April 1988.

[3] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*, pages 427–437, 1990.