CSE 5852: Lecture 17

October 31, 2016

1 Last Class

Last class we started to discuss public-key cryptography. As a reminder our goal is to allow a sender and receiver to communicate securely without having a preshared key.

This was the first time where we considered an interactive protocol. A protocol Exchange will be run between the sender and the receiver. At first we will consider a passive attacker. We will consider the notion of a transcript of the Exchange protocol denoted Transcript. We assume that the attacker sees all messages but does nothing to disrupt their transmission.

Definition 1. *A protocol* Exchange *producing keys in $\mathcal{K}$ is a passively secure key exchange protocol if for all PPT $\mathcal{A}$ there exists a negligible function such that*

$$\Pr_{k\leftarrow\mathsf{Exchange}}[\mathcal{A}(\mathsf{Transcript},k)=1]-\Pr_{k\leftarrow\mathsf{Exchange},u\leftarrow\mathcal{K}}[\mathcal{A}(\mathsf{Transcript},u)=1]|<\frac{1}{2}+\epsilon(n).$$

Definition 2. *The protocol Diffie-Hellman* Exchange *for the public parameters (p,g) is the following: S creates x and sends g^x, R creates y and sends g^y. Both S and R compute g^{xy} and output this value as the key.*

Theorem 1. *Under the decisional Diffie-Hellman assumption the Diffie-Hellman protocol is a passively secure key exchange protocol.*

2 What can go wrong?

Our adversary is under no obligation to be passive while the key exchange protocol is happening.

Attacker-in-the-middle (Often known as person-in-the-middle or man-in-the-middle.)

Suppose we have a pesky attacker who wishes to listen in on what is being said by the sender and receiver. This attacker is able to intercept messages between the sender and receiver and change them arbitrarily. We simply execute the following protocol.

1. Receive g^x from S.
2. Sample x_1 and send g^{x_1} to R.

3. Receive g^y from receiver.

4. Sample y_1 and send g^{y_1} to S.

What happens after this protocol. The sender has agreed on a key g^{xy_1} which can be computed by $\mathcal{A}$ and the receiver has agreed on a key $g^{x_1 y}$ which can be computed by $\mathcal{A}$. So the attacker knows the keys to be used by both parties. Thus, the adversary can continue to intercept (and modify) messages from S to R. Furthermore, $\mathcal{A}$ was able to generate the properly distributed values for both parties. They have no hope of being able to distinguish.

This comes back to the issue of identity, when I connected as a client I wanted to talk to `newstore.com` but had no information about who this really is. So an attacker can claim to be this person and I can't tell the difference.

This is one of the main reasons that public-key cryptography is so difficult to get right. What are valid attack models and what does identity really mean?

In the coming classes we will introduce some new tools to try and help with this problem but it remains an issue and is a major point of contention in internet design.

This is not an issue of the specific Diffie-Hellman protocol. *If at the start of the protocol no party has any information on the identity of the other, attacker in the middle is always possible.*

3 How do we establish identity?

In our running example we knew we were trying to connect to newstore.com. This wasn't enough as this information was public and the attacker could pretend to newstore.com. We need a stronger cryptographic notion of identity. We need our cryptographic identity to have two properties:

1. The creator of the identity is capable of performing some action that is computationally hard for other parties.

2. The identity can be publicly known by everyone (without subverting property 1).

This is fundamentally different than the computational hardness that we have been talking about to this point of the class. Until now we've consider functions that are possible to compute but are difficult to invert. The pseudorandom generator, discrete Log, and Diffie-Hellman protocol all had this flavor. These are collectively known as one-way functions. Anyone can compute them and no one is capable of breaking these functions. Indeed any function with kind of flavor is sufficient to build all of private key cryptography (pseudorandomfunctions, symmetric encryption, and multi-time MACs).

What we are looking for now is something different, we are looking for a party that is capable of performing an action that isn't performable by anyone else in the system. This will be our mechanism to verify their identity, if the party we're communicating with can perform this action we'll think we are talking to the intended recipient, otherwise we'll assume there is a problem.

Going forward we will consider two types of verification for this kind of identity. The first is the ability for this party to read a message that no one else

can read and the second is the ability for this party to say they sent a message in a way no one else can do.

We will spend the next few weeks working through these two tasks.

4 Public-Key Encryption

The first task we identified above is the ability to send a message to someone and have only them read it. Before we give a formal definition lets be clear about what distinguishes this from the setting of symmetric cryptography. In symmetric cryptography, we had some private key that was shared by the receiver that allowed us to communicate. Here there is some public identity that is known by all parties. However, we still want to be able to send a message to this person in a way that only they can read it.

So as before we'll have three algorithms $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$. However, they won't be the same as in the private key setting. How should they look?

Definition 3. *A tuple of algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a public-key encryption scheme if the following hold:*

1. *The generate algorithm outputs a pair of keys* $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$.
2. *The encryption algorithm takes a message and the public key* $c \leftarrow \mathsf{Enc}(pk, m)$.
3. *The decryption algorithm takes a ciphertext and the secret/private key* $m \leftarrow \mathsf{Enc}(sk, m)$.
4. *The correctness property says that decryption should always succeed:*

$$\Pr_{\mathsf{Enc},\mathsf{Dec},sk,pk}[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m] = 1.$$

With public-key encryption we want the same intuitive property of hiding that we did in private-key encryption: the adversary should learn nothing about the message from the ciphertext. The key difference is this should be true even if they know the public-key.

Definition 4. *A public-key cryptosystem for* 1*-bit messages is polynomially-secure if for all polynomial time* $\mathcal{A}$ *there exists a negligible function* $\epsilon(n)$ *such that*

$$|\Pr_{pk,sk,\mathsf{Enc}}[\mathcal{A}(pk, \mathsf{Enc}(pk, 0)) = 1] - \Pr_{pk,sk,\mathsf{Enc}}[\mathcal{A}(pk, \mathsf{Enc}(pk, 1)) = 1]| < \epsilon(n).$$

Remember we had the more complicated semantic security which hide all functions of the message.

Definition 5 (Semantic Security)**.** *[GM84] A public-key encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* semantically-secure *if for every PPT* $\mathcal{A}$ *there exists a simulator* $\mathcal{A}'$ *such that for any message distribution* M *over* $\mathcal{M}$ *and for any* $f, h : \mathcal{M} \to \{0,1\}^*$ *there exists a negligible function* $\epsilon(n)$ *such that,*

$$\Pr_{pk}[\mathcal{A}(pk, \mathsf{Enc}(pk, M), h(M)) = f(M))] - \Pr[\mathcal{A}'(h(M)) = f(M)]| < \epsilon.$$

Why doesn't $\mathcal{A}'$ receive the value pk? Its unrelated to anything else, they could generate a random pk on their own. It provides them with no additional power.

4.1 Differences and similarities with private-key encryption security

These two definitions look very similar to the private-key setting. This can be a positive thing as they designed to represent the same ideas. However, its important to understand the similarities and the differences.

Similarities

Theorem 2. *A public-key cryptosystem is semantically secure if and only if it is polynomially-secure (has indistinguishable encryptions).*

This proof is very similar to the private-key setting. We won't show this in class.

Differences

Theorem 3. *The function* Enc *must be randomized for a public-key cryptosystem to be polynomially secure.*

Theorem 4. *A public-key cryptosystem that is secure for a single message is secure for any polynomial number of messages.*

Before going into these theorems we'll give a quick example of a public-key encryption scheme. This is based on the Diffie-Hellman key exchange that we saw in the previous class.

Definition 6. *[ElG84] The El-Gamal public-key encryption scheme (we'll assume that p, g are public, they can be selected by the algorithm) is as follows:*

- $\mathsf{Gen}(p, g)$*:*

  Select x from $\mathbb{Z}_p^*$.

  Compute g^x.

  Output $pk = g^x, sk = x$.

- $\mathsf{Enc}(g^x, m \in \{0, 1\})$*:*

  Select $y \in \mathbb{Z}_p^*$ compute g^y

  If $m = 0$ send $c = (g^y, (g^x)^y)$ if $m = 1$ send $c = (g^y, g(g^x)^y = g^{xy+1})$.

- $\mathsf{Dec}(x, c)$*:*

  Parse c as $g^y = g^z$.

  Compute $(g^y)^x$ and then compute $h = (g^{xy})^{-1}$.

  If $hg^z = 1$ output 0, if $hg^z = g$ output 1.

Theorem 5. *The ElGamal public-key cryptosystem is secure under the decisional Diffie-Hellman assumption.*

References

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 10–18. Springer, 1984.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.