CSE 5852, Modern Cryptography: Foundations Fall 2016

Lecture 18, 2016

Prof. Benjamin Fuller *Scribe: Lowen Peng*

1 Overview

We will see that, for public-key encryption schemes, polynomial security is equivalent to ind-cpa security.

2 Last Class

- The attacker-in-the-middle scenario illustrates the need for a cryptographic "ID".
- One notion of public-key encryption is polynomial security.

  Definition 1. *A 3-tuple* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* ***polynomially secure*** *if* $\forall$ *PPT* $\mathcal{A}$, $\exists$ *negligible* ϵ *such that*

  $$|\Pr[\mathcal{A}(\mathrm{pk}, \mathsf{Enc}(\mathrm{pk}, 0)) = 1] - \Pr[\mathcal{A}(\mathrm{pk}, \mathsf{Enc}(\mathrm{pk}, 1)) = 1]| \leq \epsilon(n)$$

- The 1984 El Gamal encryption scheme, which works as follows.

Algorithm 1: The Gen algorithm.

Input : (p, g)
Output: $(\mathrm{pk}, \mathrm{sk})$
generate random $x \leftarrow \mathbb{Z}_p^\times$
compute $\mathrm{pk} := g^x$
compute $\mathrm{sk} := x$
return $(\mathrm{pk}, \mathrm{sk})$

Algorithm 2: The Enc algorithm.

Input : (g^x, m) where $m \in \{0, 1\}$
Output: c
generate random $y \leftarrow \mathbb{Z}_p^\times$
if $m = 0$ set $c := (g^y, g^{xy})$
else if $m = 1$ set $c := (g^y, g^{xy+1})$
return c

3 Public Key Encryption for Multiple Messages

Public-key encryption has the tremendous benefit that we don't need to worry about how many messages are being protected. If the system is secure for a single message then it works for an

arbitrary polynomial number of messages. We first need to define a multiple message notion of security.

In the past we've consider multiple message security in the setting where the adversary considers a vector of messages. This vector was implicitly chosen upfront by the adversary. Importantly, they didn't get to see ciphertexts before choosing their next plaintext. We'll have to define a more complicated experiment to model the setting where the adversary is able to select their messages adaptively.

Consider the following experiment $\mathsf{ind\text{-}cpa}^{\mathsf{Gen},\mathsf{Enc},\mathcal{A}}$:

Algorithm 3: The $\mathsf{ind\text{-}cpa}^{\mathsf{Gen},\mathsf{Enc},\mathcal{A}}$ experiment.

Output: 0 or 1
generate $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathsf{Gen}_{\mathcal{K}}(1^n)$
generate random $b \leftarrow \{0,1\}$
for *polynomially-many times* **do**
  receive $m_{i,0}, m_{i,1} \leftarrow \mathcal{A}$
  send $\mathsf{Enc}(\mathrm{pk}, m_{i,b})$
end
receive $b' \leftarrow \mathcal{A}$
if $b = b'$ output 1
else output 0

The experiment leads us to our next definition.

Definition 2. *A scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* ind-cpa ***secure*** *if* $\forall$ *PPT* $\mathcal{A}$, $\exists$ *negligible* ϵ *such that*

$$\Pr\left[\mathsf{ind\text{-}cpa}^{\mathsf{Gen},\mathsf{Enc},\mathcal{A}} = 1\right] \leq 2^{-1} + \epsilon(n)$$

Note this definition differs from polynomially secure in two important ways. First, the adversary gets to see multiple encryptions. Second they get to choose the messages both after the public-key has been specified and after seeing all of the previous messages. Somewhat surprisingly, this extra power does not help the adversary.

Now we can look at the main theorem of these notes.

Theorem 3. *A scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is polynomially secure if and only if it is also* ind-cpa *secure.*

We only show that polynomial security implies ind-cpa. Define exp-0 to be the ind-cpa experiment where, instead of randomly generating b, we fix $b := 0$. Similarly, we define exp-1 to be the case where we fix $b := 1$. Which is to say, the adversary sees, for each experiment,

exp-0	exp-1
$\mathsf{Enc}(\mathrm{pk}, m_{1,0})$	$\mathsf{Enc}(\mathrm{pk}, m_{1,1})$
$\vdots$	$\vdots$
$\mathsf{Enc}(\mathrm{pk}, m_{\ell,0})$	$\mathsf{Enc}(\mathrm{pk}, m_{\ell,1})$

Then we make the observation that if $\mathcal{A}$ breaks ind-cpa security

$$\begin{aligned}\Pr\left[\mathsf{ind\text{-}cpa}^{\mathsf{Gen},\mathsf{Enc},\mathcal{A}} = 1\right] &= 2^{-1} + 2^{-1}\left(\Pr\left[\mathsf{exp\text{-}0}^{\mathsf{Gen},\mathsf{Enc},\mathcal{A}} = 1\right] - \Pr\left[\mathsf{exp\text{-}1}^{\mathsf{Gen},\mathsf{Enc},\mathcal{A}} = 1\right]\right) \\ &\geq 2^{-1} + p(n)^{-1}\end{aligned}$$

for some polynomial p. In particular,

$$\left|\Pr\left[\mathsf{exp\text{-}0}^{\mathsf{Gen},\mathsf{Enc},\mathcal{A}}=1\right]-\Pr\left[\mathsf{exp\text{-}1}^{\mathsf{Gen},\mathsf{Enc},\mathcal{A}}=1\right]\right|\geq q(n)^{-1}$$

for some polynomial q. It suffices to show that there exists some $\mathcal{A}'$ that breaks the one-time security of the cryptosystem. For this proof its important for $\mathcal{A}'$ to have a bound on the number of queries that can be made by $\mathcal{A}$. (We can find a bound on the number of queries by running $\mathcal{A}$ by assuming different powers of n as a bound. If $\mathcal{A}$ asks for more queries than we expected we restart the experiment with n^{c+1}. This will only had a fixed multiplier to our running time and won't change whether $\mathcal{A}'$ can execute in polynomial time.)

We'll assume that $\mathcal{A}$ always makes $\ell(n)$ queries to the $\mathtt{ind}-\mathtt{cpa}$ oracle.

We define intermediate experiments $\mathsf{exp}\text{-}j$ for $1<j\leq\ell(n)$ such that for query i, where $i<j$, the adversary receives $\mathsf{Enc}(\mathrm{pk},m_{i,0})$, and for all i thereafter the adversary receives $\mathsf{Enc}(\mathrm{pk},m_{i,1})$. Which is to say, the adversary sees,

$$\begin{array}{c}\mathsf{exp\text{-}j}\\\hline\mathsf{Enc}(\mathrm{pk},m_{1,0})\\\vdots\\\mathsf{Enc}(\mathrm{pk},m_{j-1,0})\\\mathsf{Enc}(\mathrm{pk},m_{j,1})\\\vdots\\\mathsf{Enc}(\mathrm{pk},m_{\ell,1})\end{array}$$

Now we invoke the same hybrid argument as used in prior notes. The details are left to the reader.

Thus, we know that $\exists i$ such that, for some polynomial p,

$$\left|\Pr\left[\mathsf{exp\text{-}}i^{\mathsf{Gen},\mathsf{Enc},\mathcal{A}}=1\right]-\Pr\left[\mathsf{exp\text{-}}(i+1)^{\mathsf{Gen},\mathsf{Enc},\mathcal{A}}=1\right]\right|\geq\frac{1}{(\ell(n)q(n))}$$

Now consider an adversary $\mathcal{A}'$

Algorithm 4: The adversary $\mathcal{A}'$.

Input : (pk,c,i) where $c=\mathsf{Enc}(\mathrm{pk},0)$ or $c=\mathsf{Enc}(\mathrm{pk},1)$

for $j<i$ **do**
  invoke $\mathcal{A}$ on pk
  receive $(m_{j,0},m_{j,1})$
  send $\mathsf{Enc}(\mathrm{pk},m_{j,0})$
end
if $j=i$ **then**
  send c
end
for $i<j\leq\ell(n)$ **do**
  invoke $\mathcal{A}$ on pk
  receive $(m_{j,0},m_{j,1})$
  send $\mathsf{Enc}(\mathrm{pk},m_{j,1})$
end

We won't formally show that $\mathcal{A}'$ beats ind-cpa security but the basic idea is they are either receiving an encryption of 0 or 1 and since the underlying $\mathcal{A}$ has different outputs in the two cases so will $\mathcal{A}'$.

4 Is public-key encryption good enough to create a secure channel?

We've now shown how we can achieve public-key encryption. How can we use this to achieve a secure channel between the two parties?

4.1 Hybrid Encryption

A simple way of achieving this is to have one party send the other a symmetric encryption key (encrypted under the public-key encryption) and then encrypt their entire message using a symmetric encryption scheme (possible made secure against modification using a MAC algorithm).

Let $(PGen, PEnc, PDec)$ be a public-key cryptosystem. And let $(SGen, SEnc, SDec)$ be a private-key cryptosystem.

Suppose we have a S and receiver R with public keys pk_S and pk_R respectively. Use the following mechanism to have a secure channel between the two parties:

1. S has a message m to send.
2. Run $k \leftarrow SGen(1^n)$.
3. Encrypt $c_m = SEnc(k, m)$.
4. Encrypt $c_k = PEnc(pk_R, k)$.
5. Send c_k, c_m

The receiver can perform similar actions when they are sending their message. This technique is known as hybrid encryption. We use the public-key encryption scheme to send an encryption key for a symmetric key scheme and then encrypt our message with the symmetric scheme (we can use a symmetric scheme with a MAC).

4.2 Public-Key encapsulation

We can take the above protocol a step further. Rather than the two parties generating a new key for each message they can just use the public-key encryption to agree upon a shared key. So a new key agreement protocol becomes $k \leftarrow SGen(1^n)$ and then S sends $c = PEnc(pk_R, k)$. Then both parties send using that key on the symmetric channel.

4.3 The issue of malleability

Remember our goal in introducing identity was to prevent an active attacker from being able to interrupt the key exchange protocol. Using the Diffie-Hellman protocol, the attacker was able to cause the two parties to agree on separate keys and know the keys for both parties.

Because we are now using public-key encryption we have gained the property that the attacker cannot learn the key we are sending. This is definitely an improvement. However, we have no guarantee that the attacker can not modify c and cause the recipient to get a different key.

Suppose we are at the situation where the parties have keys k and k' that are different? Is there an easy way for the parties to detect this?

Suppose we use a good MAC to try and detect the difference.

So after sending c. The sender or receiver creates a random value r and sends $MAC(k, r)$. We know by the guarantees of the MAC that the attack cannot change r to a value r' in the setting where both parties share k. However it might be possible to change the MAC value in a way that makes it work for a different key. This was never a requirement of the protocol.

Really to bootstrap we need the ability to send a message where a change can be detected. This leads us to the notion of public-key signatures. These are messages that can only be created by a single individual. We'll cover these in detail next week.