Modern Cryptography

600.442

Lecture #16

Dr. Christopher Pappacena

Fall 2013

Public-Key Encryption

A *public-key encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is given by:

- $\mathsf{Gen}(n)$ outputs a *public key* $\mathtt{pk}$ and a *secret key* $\mathtt{sk}$.

- Enc takes as input the public key and a message m and returns a ciphertext c. We write $c \leftarrow \mathsf{Enc}_{\mathtt{pk}}(m)$.

- Dec takes as input the secret key and a ciphertext c and returns either a message m or a decryption failure $\perp$. We write $m = \mathsf{Dec}_{\mathtt{sk}}(c)$.

We require that $\Pr[\mathsf{Dec}_{\mathtt{sk}}(\mathsf{Enc}_{\mathtt{pk}}(m)) \neq m] = \mathtt{negl}(n)$ for some negligible function of n.

Remarks

- The public key is made available so that *anyone* can encrypt messages.

- We allow a negligible probability of decryption failure.

- We generally need a method for encoding messages as elements of a group. For RSA, we can encode a nonzero message of $\leq n-1$ bits by viewing it as an integer in $\{1, \ldots, N-1\}$.

- For El Gamal encryption this encoding is more complicated.

The Indistinguishability Experiment

- Gen is run to produce $\mathtt{pk}$ and $\mathtt{sk}$.
- $\mathcal{A}$ is given $\mathtt{pk}$ and produces two messages m_0, m_1 of the same length.
- $b \leftarrow \{0, 1\}$ is chosen at random and $\mathcal{A}$ is given $c \leftarrow \mathsf{Enc}_{\mathtt{pk}}(m_b)$.
- $\mathcal{A}$ outputs a bit b'. We write $\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1$ if $b = b'$ and say that $\mathcal{A}$ *succeeds*; otherwise $\mathcal{A}$ *fails*.

Definition: A public-key encryption scheme Π has indistinguishable encryptions in the presence of an eavesdropper if, for all PPT adversaries $\mathcal{A}$, we have

$$\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \mathtt{negl}(n)$$

for some negligible function $\mathtt{negl}$.

Remark: There is *no* notion of "perfect secrecy" for a public-key encryption schems. A computationally unbounded adversary can recover the message m from c with probability 1 (HW Problem).

Public-Key Encryption and CPA Security

In the public-key setting, the adversary is given access to the public key $\mathtt{pk}$. This means two things:

- Enc *must* be randomized. Otherwise, $\mathcal{A}$ just computes $c_b = \mathsf{Enc}_{\mathtt{pk}}(m_b)$ for $b = 0, 1$ and easily succeeds.

- $\mathcal{A}$ has access to an encryption oracle and so is able to mount a chosen-plaintext attack.

Proposition: If a public-key encryption scheme Π has indistinguishable encryptions in the presence of an eavesdropper, then Π is also CPA secure.

Multiple Encryptions

We can define an experiment $\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)$, where the adversary $\mathcal{A}$ produces a pair of message *vectors* $M_0 = (m_0^1, \ldots, m_0^t)$ and $M_1 = (m_1^1, \ldots, m_1^t)$ for some t.

The game is played the same way, with $\mathcal{A}$ receiving a ciphertext vector C_b. $\mathcal{A}$ returns b' and succeeds if $b = b'$.

We define security in the presence of multiple encryptions in the obvious way.

One For All and All For One

Theorem: If a public key scheme Π has indistinguishable encryptions in the presence of an eavesdropper, then it has indistinguishable multiple encryptions in the presence of an eavesdropper.

The proof uses a hybrid argument.

Proof of Theorem

In the experiment $\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)$, $\mathcal{A}$ selects two plaintext vectors M_0 and M_1 of length t.

For $0 \le i \le t$, define the ciphertext vector $C^{(i)}$ by

$$C^{(i)} = (\mathsf{Enc}_{\mathtt{pk}}(m_0^1), \ldots, \mathsf{Enc}_{\mathtt{pk}}(m_0^i), \mathsf{Enc}_{\mathtt{pk}}(m_1^{i+1}), \ldots, \mathsf{Enc}_{\mathtt{pk}}(m_1^t)).$$

In words, $C^{(i)}$ is an encryption of the first i plaintexts of M_0 followed by the last $t-i$ plaintexts of M_1.

As we range over all possible randomizations of Enc, each $C^{(i)}$ defines a *distribution* on vectors with t ciphertexts.

Proof of Theorem, II

Now let $\mathcal{A}'$ be an adversary which uses $\mathcal{A}$ as a subroutine:

- $\mathcal{A}'$ gives $\mathtt{pk}$ to $\mathcal{A}$ and receives M_0 and M_1.
- $\mathcal{A}'$ chooses $i \leftarrow \{1, \ldots, t\}$, outputs m_0^i, m_1^i, and receives c^i.
- $\mathcal{A}'$ computes $c^j \leftarrow \mathsf{Enc}_{\mathtt{pk}}(m_0^j)$ for $j < i$ and $c^j \leftarrow \mathsf{Enc}_{\mathtt{pk}}(m_1^j)$ for $j > i$.
- $\mathcal{A}'$ gives $(c^1, \ldots, c^t)$ to $\mathcal{A}$ and returns the bit b' returned by $\mathcal{A}$.

Proof of Theorem, III

If $b = 0$, then the ciphertext that $\mathcal{A}'$ gives $\mathcal{A}$ is $C^{(i)}$. We have:

$$\begin{aligned}\Pr[\mathcal{A}'(n) = 0 | b = 0] &= \sum_{j=1}^{t} \Pr[\mathcal{A}'(n) = 0 | b = 0 \wedge i = j] \times \Pr[i = j] \\ &= \frac{1}{t} \sum_{j=1}^{t} \Pr[\mathcal{A}(C^{(j)}) = 0].\end{aligned}$$

If $b = 1$, then the ciphertext is $C^{(i-1)}$ and

$$\Pr[\mathcal{A}'(n) = 1 | b = 1] = \frac{1}{t} \sum_{j=0}^{t-1} \Pr[\mathcal{A}(C^{(j)}) = 1].$$

Proof of Theorem, IV

Combining these gives

$$\begin{aligned}
\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}',\Pi}(n) = 1] &= \frac{1}{2}\Pr[\mathcal{A}'(n) = 0 | b = 0] + \frac{1}{2}\Pr[\mathcal{A}'(n) = 1 | b = 1] \\
&= \frac{1}{2t}\left(\sum_{j=1}^{t} \Pr[\mathcal{A}(C^{(j)}) = 0] + \sum_{j=0}^{t-1} \Pr[\mathcal{A}(C^{(j)}) = 1]\right) \\
&= \frac{t-1}{2t} + \frac{1}{2t}\Pr[\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n) = 1].
\end{aligned}$$

If the advantages of $\mathcal{A}$ and $\mathcal{A}'$ are $\epsilon(n)$ and $\epsilon'(n)$ then this gives

$$\epsilon(n) = t \cdot \epsilon'(n).$$

Recap

- Indistinguishable encryptions in the presence of an eavesdropper implies indistinguishable *multiple* encryptions in the presence of an eavesdropper.

- We can bootstrap a fixed-length public-key system into one for arbitrary-length messages.

- Indistinguishability implies CPA-security for public-key encryption. As a result, any secure public-key encryption scheme must have *randomized* encryptions.

- "Textbook" RSA doesn't use randomization, so is insecure!

Padded RSA

Let $\ell(n)$ be a function with $\ell(n) \leq 2n-2$. To encrypt $m \in \{0,1\}^{\ell(n)}$, choose a random $r \leftarrow \{0,1\}^{\|N\|-\ell(n)-1}$ and set

$$c = (r||m)^e \pmod{N}.$$

To decrypt, let $\tilde{m} = c^d \pmod{N}$ and set m equal to the low $\ell(n)$ bits of $\tilde{m}$.

Theorem: If $\ell(n) = O(\log n)$ and the RSA problem is hard relative to GenRSA, then this gives a CPA-secure public-key encryption scheme.

The proof uses the fact that the low-order bits give hard-core predicates for RSA.

PKCS #1 v1.5

Let k be the size of N in bytes. Messages m range from 1 to $k-11$ bytes long.

The encryption of an s-byte message m is

$$(00000000||00000010||r||00000000||m)^e \pmod{N},$$

where r is a random string of $k-s-3$ nonzero bytes.

This is believed to be CPA-secure but no proof is known based on the RSA assumption. It is known to *not* be CCA-secure.

El Gamal Encryption

In 1985, El Gamal introduced a public-key encryption scheme whose security is based on the DDH problem.

Let $\mathcal{G}$ be a group generation algorithm for which the DDH problem is hard.

- Run $\mathcal{G}(n)$ to obtain (G, q, g). Choose $x \leftarrow \mathbb{Z}_q$ and set $h = g^x$. Set $\texttt{pk} = (G, q, g, h)$ and $\texttt{sk} = (G, q, g, x)$.

- Given a message $m \in G$, choose $y \leftarrow \mathbb{Z}_q$ and set $c = (g^y, h^y m)$.

- Given $c = (c_1, c_2)$, decrypt by setting $m = c_2 / c_1^x$ $(= c_2 \circ c_1^{-x})$.

Theorem: If the DDH problem is hard relative to $\mathcal{G}$, then El Gamal encryption scheme Π has indistinguishable encryptions in the presence of an eavesdropper and is CPA-secure.

Proof: Let $\mathcal{A}$ be an adversary who can break Π with advantage $\epsilon(n)$.

Define $\widetilde{\mathsf{Enc}}_{\mathrm{pk}}(m)$ to be $(g^y, g^z m)$ for random $y, z \leftarrow \mathbb{Z}_q$.

If $\widetilde{\mathsf{Enc}}_{\mathrm{pk}}$ is used in place of $\mathsf{Enc}_{\mathrm{pk}}$ then $\mathcal{A}$ will succeed with probability $1/2$ since no information about m is revealed.

Proof, Continued

We design a distinguisher D for the DDH problem. Recall that D is given (G, q, g, g^x, g^y, h) and needs to decide if $h = g^{xy}$.

- D gives $\texttt{pk} = (G, q, g, g^x)$ to $\mathcal{A}$ and receives m_0, m_1.

- D chooses b, gives $c = (g^y, hm_b)$ to $\mathcal{A}$, and receives b'.

- If $b' = b$ then D returns 1, otherwise D returns 0.

Proof, Continued

If $h \neq g^{xy}$, then $\mathcal{A}$ returns 0 and 1 with probability 1/2 each. So D returns 1 with probability 1/2 in this case.

If $h = g^{xy}$, then $\mathcal{A}$ returns b with probability $1/2 + \epsilon(n)$ and so D returns 1 with probability $1/2 + \epsilon(n)$ in this case.

Hence,

$$|\Pr[D(h = g^{xy}) = 1] - \Pr[D(h \neq g^{xy}) = 1]| = \epsilon(n).$$

Since the DDH problem is hard for $\mathcal{G}$, $\epsilon(n)$ is negligible.

Factoring and One-Way Functions

Let $t(n)$ be the maximum number of random bits needed by GenMod(n) to produce (p, q, N).

We define a function $f : \{0,1\}^{t(n)} \to \{0,1\}^{2n}$ as follows:

- Run GenMod(n) using input $x \in \{0,1\}^{t(n)}$ in place of the random bits.
- Given the output (p, q, N) of GenMod, let $f(x) = N$.

If f can be inverted, then the input to f can be used to find the factors of N. So if factoring is hard relative to GenMod, then f is a one-way function.

The RSA Problem and One-Way Permutations

Let GenRSA(n) $= (N, e, d)$. Then the map

$$f_{e,N} : x \mapsto x^e \pmod N$$

is a *permutation* of the set $\mathbb{Z}_N^*$, with inverse $f_{d,N}$.

The problem of computing x given $y = f_{e,N}(x)$ is exactly the RSA problem with input y.

So if the RSA problem is hard relative to GenRSA, then $f_{e,N}$ is (morally) a one-way permutation.

In order to make this idea fit the general framework of one-way functions we have to be a bit fussy. Details are in the book.

Hash Functions

Define a fixed-length hash function H as follows:

- Run $\mathcal{G}(n)$ to obtain (G, q, g), select $h \leftarrow G$, and set $s = (G, q, g, h)$.

- Given input $x = (x_1, x_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$, ouptut $H^s(x) = g^{x_1} h^{x_2}$.

H looks like a hash function since it reduces two elements of a cyclic group of order q to a single element of a cyclic group of order q.

For it to actually compress its output we need elements of G to be encoded by no more than $2n - 2$ bits, or use a *randomness extractor*.

The Discrete Logarithm Problem and Hash Functions

Theorem: If the discrete logarithm problem is hard relative to $\mathcal{G}$ and H compresses its input, then H is a collision-resistant hash function.

Proof: Suppose $\mathcal{A}$ can invert H. Given $s = (G, q, g, h)$, call $\mathcal{A}$ as a subroutine with seed s and receive x, x'.

Write $x = (x_1, x_2)$ and $x' = (x'_1, x'_2)$ and set $y = (x_1 - x'_1)/(x_2 - x'_2) \pmod{q}$.

If $H^s(x) = H^s(x')$, then $y = \log_g h$.

The above can be turned into a formal security proof – see Theorem 7.73 of Katz and Lindell.