Modern Cryptography

600.442

Lecture #17

Dr. Christopher Pappacena

Fall 2013

Announcements

- Homework 8 due on Tuesday, November 19.

- Exam 2 on Thursday, November 21.

- Exam will cover Chapters 6, 7, 9, and 10.

Last Time

- RSA and El Gamal Encryption
- Multi-Message and CPA Security of Public-Key Cryptography
- Connections between factoring and the discrete log problem to one way functions, one way permutations, and secure hash functions.

Hybrid Encryption

In general, public-key encryption is more computationally expensive than private-key encryption.

According to the book, a reasonable estimate for the relative efficiency of private-key to public-key encryption is a factor of 10^6!

We would like to combine the functionality of public-key encryption with the efficiency of private-key encryption, particularly for long messages.

Hybrid Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme and let $\Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ be a private-key encryption scheme.

We define a *hybrid* scheme $\Pi^{\mathsf{hyb}} = (\mathsf{Gen}^{\mathsf{hyb}}, \mathsf{Enc}^{\mathsf{hyb}}, \mathsf{Dec}^{\mathsf{hyb}})$ as follows.

- $\mathsf{Gen}^{\mathsf{hyb}}(n)$ runs $\mathsf{Gen}(n)$ to obtain keys $\mathtt{pk}$ and $\mathtt{sk}$.
- Run $\mathsf{Gen}'(n)$ to obtain a key $k \leftarrow \{0,1\}^n$. Set $c_1 \leftarrow \mathsf{Enc}_{\mathtt{pk}}(k)$, $c_2 \leftarrow \mathsf{Enc}'_k(m)$, and $(c_1, c_2) \leftarrow \mathsf{Enc}^{\mathsf{hyb}}_{\mathtt{pk}}(m)$.
- To decrypt $c = (c_1, c_2)$, set $k = \mathsf{Dec}_{\mathtt{sk}}(c_1)$ and $m = \mathsf{Dec}'_k(c_2)$.

Note that Π^{hyb} is a *public-key scheme*, even though it uses a private-key scheme to do “most” of the encryption.

Theorem: If Π is a CPA-secure public-key encryption scheme and Π' has indistinguishable encryptions in the presence of an eavesdropper, then Π^{hyb} is a CPA-secure public-key encryption scheme.

Outline of Proof

An adversary $\mathcal{A}$ cannot distinguish:

- $(\mathtt{pk}, \mathsf{Enc}_{\mathtt{pk}}(k), \mathsf{Enc}'_k(m_0))$ from $(\mathtt{pk}, \mathsf{Enc}_{\mathtt{pk}}(0^n), \mathsf{Enc}'_k(m_0))$.

- $(\mathtt{pk}, \mathsf{Enc}_{\mathtt{pk}}(0^n), \mathsf{Enc}'_k(m_0))$ from $(\mathtt{pk}, \mathsf{Enc}_{\mathtt{pk}}(0^n), \mathsf{Enc}'_k(m_1))$.

- $(\mathtt{pk}, \mathsf{Enc}_{\mathtt{pk}}(0^n), \mathsf{Enc}'_k(m_1))$ from $(\mathtt{pk}, \mathsf{Enc}_{\mathtt{pk}}(k), \mathsf{Enc}'_k(m_1))$.

By "transitivity of indistinguishability", Π^{hyb} has indistinguishable encryptions and hence is CPA-secure.

Adversary $\mathcal{A}_1$

Let $\mathcal{A}^{\mathsf{hyb}}$ be an adversary that can break Π^{hyb} with advantage $\epsilon(n)$.

$\mathcal{A}_1$ will be an adversary which calls $\mathcal{A}^{\mathsf{hyb}}$ as a subroutine:

- $\mathcal{A}_1$ is given `pk`, runs $\mathsf{Gen}'(n)$ to produce k, gives the messages $k, 0^n$, and receives c.
- $\mathcal{A}_1$ calls $\mathcal{A}^{\mathsf{hyb}}$ as a subroutine and receives m_0, m_1.
- $\mathcal{A}_1$ gives $(c, \mathsf{Enc}'_k(m_0))$ to $\mathcal{A}^{\mathsf{hyb}}$, and returns the bit b' returned by $\mathcal{A}^{\mathsf{hyb}}$.

Success Probability for $\mathcal{A}_1$

We have

$$\begin{aligned}\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}_1,\Pi}(n) = 1] = &\frac{1}{2}\Pr[\mathcal{A}^{\mathsf{hyb}}(\mathsf{Enc}_{\mathtt{pk}}(k), \mathsf{Enc}'_k(m_0)) = 0] \\ &+ \frac{1}{2}\Pr[\mathcal{A}^{\mathsf{hyb}}(\mathsf{Enc}_{\mathtt{pk}}(0^n), \mathsf{Enc}'_k(m_0)) = 1].\end{aligned}$$

Since Π is CPA-secure, this success probability is $\leq 1/2 + \mathtt{negl}_1(n)$ for some negligible function $\mathtt{negl}_1$.

Adversary $\mathcal{A}'$

Next consider an adversary $\mathcal{A}'$ which uses $\mathcal{A}^{\mathsf{hyb}}$ as a subroutine:

- $\mathcal{A}'$ runs Gen(n) to produce keys $\mathtt{pk}, \mathtt{sk}$.
- $\mathcal{A}'$ runs $\mathcal{A}^{\mathsf{hyb}}$ with public key $\mathtt{pk}$ and receives m_0, m_1.
- $\mathcal{A}'$ outputs m_0, m_1 and receives c.
- $\mathcal{A}'$ gives $(\mathsf{Enc}_{\mathtt{pk}}(0^n), c)$ to $\mathcal{A}^{\mathsf{hyb}}$ and returns the bit b' output by $\mathcal{A}^{\mathsf{hyb}}$.

Success Probability for $\mathcal{A}'$

We have

$$\begin{aligned}\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A}',\Pi'}(n)=1] = \frac{1}{2}\Pr[\mathcal{A}^{\mathsf{hyb}}(\mathsf{Enc}_{\mathtt{pk}}(0^n),\mathsf{Enc}'_k(m_0))=0] \\ +\frac{1}{2}\Pr[\mathcal{A}^{\mathsf{hyb}}(\mathsf{Enc}_{\mathtt{pk}}(0^n),\mathsf{Enc}'_k(m_1)=1)].\end{aligned}$$

Since Π' has indistinguishable encryptions in the presence of an eavesdropper, this success probability is $\leq 1/2+\mathtt{negl}'(n)$ for some negligible function $\mathtt{negl}'$.

Adversary $\mathcal{A}_2$

Finally, consider the adversary $\mathcal{A}_2$ which uses $\mathcal{A}^{\mathsf{hyb}}$ as a subroutine as follows:

- $\mathcal{A}_2$ is given $\mathtt{pk}$, runs $\mathsf{Gen}'(n)$ to produce k, gives the messages $0^n, k$, and receives c.
- $\mathcal{A}_2$ calls $\mathcal{A}^{\mathsf{hyb}}$ and receives m_0, m_1.
- $\mathcal{A}_2$ gives $(c, \mathsf{Enc}'_k(m_1))$ to $\mathcal{A}^{\mathsf{hyb}}$ and returns the bit b' returned by $\mathcal{A}^{\mathsf{hyb}}$.

Success Probability for $\mathcal{A}_2$

We have

$$\begin{aligned}\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}_2,\Pi}(n)=1] = \frac{1}{2}\Pr[&\mathcal{A}^{\mathsf{hyb}}(\mathsf{Enc}_{\mathtt{pk}}(0^n),\mathsf{Enc}'_k(m_1))=0] \\ &+\frac{1}{2}\Pr[\mathcal{A}^{\mathsf{hyb}}(\mathsf{Enc}_{\mathtt{pk}}(k),\mathsf{Enc}'_k(m_1)=1)].\end{aligned}$$

Since Π is CPA-secure, this success probability is $\leq 1/2+\mathtt{negl}_2(n)$ for some negligible function $\mathtt{negl}_2$.

Putting it Together

Summing our three inequalities gives

$$\begin{aligned}\frac{3}{2} + \texttt{negl}(n) \geq \frac{1}{2}(&\Pr[\mathcal{A}^{\mathsf{hyb}}(\mathsf{Enc}_{\texttt{pk}}(k), \mathsf{Enc}'_k(m_0)) = 0] \\ &+ \Pr[\mathcal{A}^{\mathsf{hyb}}(\mathsf{Enc}_{\texttt{pk}}(0^n), \mathsf{Enc}'_k(m_0)) = 1] \\ &+ \Pr[\mathcal{A}^{\mathsf{hyb}}(\mathsf{Enc}_{\texttt{pk}}(0^n), \mathsf{Enc}'_k(m_0)) = 0] \\ &+ \Pr[\mathcal{A}^{\mathsf{hyb}}(\mathsf{Enc}_{\texttt{pk}}(0^n), \mathsf{Enc}'_k(m_1)) = 1] \\ &+ \Pr[\mathcal{A}^{\mathsf{hyb}}(\mathsf{Enc}_{\texttt{pk}}(0^n), \mathsf{Enc}'_k(m_1)) = 0] \\ &+ \Pr[\mathcal{A}^{\mathsf{hyb}}(\mathsf{Enc}_{\texttt{pk}}(k), \mathsf{Enc}'_k(m_1)) = 1]).\end{aligned}$$

Here $\texttt{negl}(n) = \texttt{negl}_1(n) + \texttt{negl}'(n) + \texttt{negl}_2(n)$ is negligible.

Cleaning Up

Simplifying the right hand side in the previous slide gives

$$\begin{aligned}\frac{3}{2} + \mathtt{negl}(n) \geq{}& 1 + \frac{1}{2}(\Pr[\mathcal{A}^{\mathsf{hyb}}(\mathsf{Enc}_{\mathtt{pk}}(k), \mathsf{Enc}'_k(m_0)) = 0] \\ &+ \Pr[\mathcal{A}^{\mathsf{hyb}}(\mathsf{Enc}_{\mathtt{pk}}(k), \mathsf{Enc}'_k(m_1)) = 1])\end{aligned}$$

So,

$$\frac{1}{2} + \mathtt{negl}(n) \geq \frac{1}{2} + \epsilon(n)$$

and $\epsilon(n)$ is negligible.

CCA Security

CCA indistinguishability for public-key encryption is defined analogously to the private-key setting: After receiving the challenge ciphertext, $\mathcal{A}$ has access to a decryption oracle for any ciphertext *except* c.

The formal definition is in Katz and Lindell, p. 371.

Definition: We say that Π is *CCA-secure* if

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

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

Homomorphic Encryption and Malleability

"Textbook RSA" and El Gamal encryption are both *homomorphic*, which means that if $c_1 = \mathsf{Enc}_{\mathtt{pk}}(m_1)$ and $c_2 = \mathsf{Enc}_{\mathtt{pk}}(m_1)$, then $c_1 c_2 = \mathsf{Enc}_{\mathtt{pk}}(m_1 m_2)$.

For textbook RSA, this says

$$c_1 c_2 = m_1^e m_2^e = (m_1 m_2)^e \pmod{N}.$$

For El Gamal, we mean coordinate-wise multiplication:

$$(g^{y_1}, h^{y_1} m_1) \cdot (g^{y_2}, h^{y_2} m_2) = (g^{y_1+y_2}, h^{y_1+y_2} m_1 m_2).$$

This *ciphertext malleability* leaves both systems open to chosen ciphertext attacks.

Example: El Gamal

- Suppose $\mathcal{A}$ generates $m_0, m_1 \in G$ and receives back $c_b = (g^y, h^y m_b)$.

- $\mathcal{A}$ chooses a random $m \in G$ and encrypts it to make $c = (g^z, h^z m)$.

- Now $\mathcal{A}$ creates the ciphertext $c' = (g^{y+z}, h^{y+z} m_b m)$ and queries the decryption oracle.

- $\mathcal{A}$ computes $m_b = (m_b m)/m$ and succeeds with probability 1.

Padded RSA

Padding RSA such as in PKCS #1 v1.5 eliminates the homomorphic property – the product of two valid messages is not likely to be a valid message.

However, an adversary can deduce information about the plaintext by observing if chosen ciphertexts decrypt succeessfully or return $\perp$.

A different padding scheme called OAEP (Optimal Asymmetric Encryption Padding) is CCA-secure in the *random oracle model* (Chapter 13).

Trapdoor Permutations

Roughly, a *trapdoor permutation* is a permutation f together with a "trapdoor" value `td`:

- Without knowledge of `td`, f is difficult to invert; i.e. f is a one-way permutation.

- Given `td`, it is possible to invert f in polynomial time.

If f is a trapdoor permutation with hardcore predicate hc, then we can construct a secure public-key system from f.

Public-Key Encryption from Trapdoor Permutations

Define Π as follows.

- We set $\texttt{pk} = (f, \mathsf{hc})$ and $\texttt{sk} = \texttt{td}$.
- To encrypt a bit b, choose $x \leftarrow \{0,1\}^n$ and set
$$\mathsf{Enc}_{\texttt{sk}}(b) = (f(x), \mathsf{hc}(x) \oplus b).$$
- To decrypt (y, c), use $\texttt{td}$ to invert $y = f(x)$ and set $b = c \oplus \mathsf{hc}(x)$.

Theorem: If f is a trapdoor permutation then Π is CPA-secure.

The idea of the proof is to note that an adversary who can break the system is computing hc(x) from $f(x)$ with some advantage. Since f is one-way and hc is a hardcore predicate, this advantage is negligible.

The hardest part is giving a definition of "trapdoor permutation" that varies with the security parameter.

Definition: A *trapdoor permutation* is a tuple $(\mathsf{Gen}, \mathsf{Samp}, f, \mathsf{Inv})$

- $\mathsf{Gen}(n)$ outputs $(I, \mathtt{td})$, where I specifies the doman D_I and range $R_I = D_I$ of the function f_I.
- $\mathsf{Samp}(n)$ samples uniformly from D_I.
- $f_I : D_I \to D_I$ is a one-way permutation.
- $\mathsf{Inv}_{\mathtt{td}}(y)$ computes the inverse of y, given the trapdoor secret: $\mathsf{Inv}_{\mathtt{td}}(f_I(x)) = x$ for all $(I, \mathtt{td})$ and $x \in D_I$.

Even though the construction only encrypts a single bit, we know that CPA-security implies indistinguishable multiple encryptions. So we can iterate the construction to encrypt arbitrary length messages.

Chapter 12: Digital Signatures

In their seminal paper, Diffie and Hellman note that a public-key encryption system can be used to provide a *digitial signature* – a message tag that only one person can produce, but anyone can verify:

If m is a message, it is signed by computing $t = \mathsf{Dec}_{\mathtt{sk}}(m)$, where Dec denotes the decryption of the "ciphertext" m with the user's secret key.

Now, anyone can verify the validity of the message with the user's *public* key: $m = \mathsf{Enc}_{\mathtt{pk}}(t)$.

For this to work, encryption must be a "two-sided" inverse to decryption. This is true of both RSA and El Gamal encryption.

While this intuition is still used to popularly describe digital signatures, it predates modern cryptography.

The formal definition of digital signature will show that it is not quite as simple as “reversing” public-key encryption.

Nevertheless, it is closely related and once again illustrates Diffie and Hellman’s insight.

Digital Signatures Defined

A *signature scheme* is a triple of PPT algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$.

- $\mathsf{Gen}(n)$ produces public and secret keys $\mathtt{pk}$ and $\mathtt{sk}$.
- Given a message $m \in \{0,1\}^*$, $\sigma \leftarrow \mathsf{Sign}_{\mathtt{sk}}(m)$ is the signature of m.
- Given a pair (m, σ), $\mathsf{Verify}_{\mathtt{pk}}(m, \sigma)$ returns a bit b. If $b = 1$ then σ is *valid*, otherwise it is *invalid*.

We require that $\mathsf{Verify}_{\mathtt{pk}}(m, \mathsf{Sign}_{\mathtt{sk}}(m)) = 1$, except possibly with negligible probability.

Uses of Digital Signatures

- Non-repudiation: like a physical signature, a party can attest to a digital document by digitally signing it.

- Message integrity: By signing a message, a party can prove that the message originated from him or her.
  - Frequently-used instance: Signing software updates.

- Digital signatures are an important part of *public-key infrastructure*, where trusted parties attest to the validity of a public key by signing it.

Digital Signatures vs. MACs

Digital signatures offer similar functionality to MACs. The main difference is that *anyone* can verify a digital signature but only the *intended recipient* can verify a MAC.

This has its benefits but has drawbacks as well:

- Digital signatures are more expensive to compute and verify than MACs.
- In some cases, the sender might only want the intended recipient to be able to authenticate the message.

In short, there is need for both primitives in different cryptographic applications.

The Signature Experiment

- Gen(n) is run to obtain signing keys $\mathtt{pk}, \mathtt{sk}$.
- $\mathcal{A}$ is given oracle access to Sign$_{\mathtt{sk}}(\cdot)$ and outputs (m, σ).
- We say that SigForge$_{\mathcal{A},\Pi}(n) = 1$ if Verify$_{\mathtt{pk}}(m, \sigma) = 1$ and $\mathcal{A}$ did not query the oracle with m. We also say that $\mathcal{A}$ *suceeds* or *forges* the signature.

Definition: A signature scheme $\Pi = (\text{Gen}, \text{Sign}, \text{Verify})$ is *existentially unforgeable under an adaptive chosen-message attack* if, for every PPT adversary $\mathcal{A}$, there is a negligible function $\mathtt{negl}$ such that

$$\Pr[\text{SigForge}_{\mathcal{A},\Pi}(n) = 1] \leq \mathtt{negl}(n).$$

Insecurity of “Textbook” RSA

Just as textbook RSA is insecure as a public-key algorithm (relative to modern notions of security), it is also insecure as a digital signature algorithm.

Given a message m, an adversary $\mathcal{A}$ can request the signature of m_1 and $m_2 = m_1^{-1}m$. If the signatures are σ_1 and σ_2, then

$$\sigma_1\sigma_2 = m_1^d m_2^d = (m_1 m_2)^d = m^d \pmod{N}$$

so $\sigma = \sigma_1\sigma_2$ is a valid signature for m.

There is even a “no message” attack: $\mathcal{A}$ chooses σ and sets $m = \sigma^e \pmod{N}$. Then σ is a valid signature for m!

Hashed RSA

Fix a collision-resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_N^*$. To sign a message m, compute

$$\sigma = H(m)^d \pmod{N}.$$

To verify (m, σ), compute $H(m)$ and check if $\sigma^e = H(m) \pmod{N}$.

There is no known function H for which hashed RSA can be proven to be secure. However, if we model H as a *random* function, then it is possible to prove security of hashed RSA.

This *random oracle* model is discussed in Chapter 13.

Hash and Sign

The hashed RSA construction doesn't just make finding collisions harder – it also allows the user to sign messages of arbitary length.

Let Π be a fixed-length signature scheme and let H be a hash function. Construct Π′ as follows.

- Gen′(n) runs Gen(n) to generate keys $\mathtt{pk}, \mathtt{sk}$ and also generates a seed s for H. Set $\mathtt{pk}' = (\mathtt{pk}, s)$ and $\mathtt{sk}' = (\mathtt{sk}, s)$.

- $\mathsf{Sign}'_{\mathtt{sk}'}(m) = \mathsf{Sign}_{\mathtt{sk}}(H^s(m))$.

- $\mathsf{Verify}'_{\mathtt{pk}'}(m, \sigma) = \mathsf{Verify}_{\mathtt{pk}}(H^s(m), \sigma)$.

Theorem: If Π is existentially unforgeable under a chosen-message attack and H is collision-resistant, then Π′ is existentially unforgeable under a chosen-message attack.

Proof Outline: Suppose an adversary $\mathcal{A}$ forges the message (m, σ) for Π′. Then he has forged the message $(H^s(m), \sigma)$ for Π. One of two things has happened:

- $\mathcal{A}$ has forged the message $H^s(m)$, which he has never seen before. This violates security of Π.
- $\mathcal{A}$ has seen $H^s(m) = H^s(m')$ before. This means that $\mathcal{A}$ has found a collision in H.

Since both of these are hard, Π is secure.