COS433/Math 473: Cryptography

Mark Zhandry
Princeton University
Spring 2017
Last Time

Exchanging Keys
Public Key Distribution
Public Key Distribution
Public Key Distribution
Public Key Distribution
Public Key Distribution

Pair of interactive algorithms $A(\lambda), B(\lambda)$

Correctness:

$$\text{Pr}[o_A = o_B: (\text{Trans}, o_A, o_B) \leftarrow (A, B)(\lambda, \lambda)] = 1$$

Shared key is $k := o_A = o_B$

- Define $(\text{Trans}, k) \leftarrow (A, B)(\lambda)$

Security: $(\text{Trans}, k)$ is computationally indistinguishable from $(\text{Trans}, k')$ where $k' \leftarrow K$
Key Distribution from RSA

\[ p, q \] random primes

\[ N = pq \]
Key Distribution from RSA

\( p, q \) random primes

\( N = pq \)

\( N \leftarrow \mathbb{Z}_N^* \)

\( y \leftarrow x^3 \mod N \)

\( x \leftarrow x \mod N \)
Key Distribution from RSA

\[ p, q \text{ random primes} \]

\[ N = pq \]

\[ x = y^{1/3} \mod N \]

\[ x \leftarrow \mathbb{Z}_N^* \]

\[ y \leftarrow x^3 \mod N \]

\[ x \]
Analysis

$x$ uniquely defined as long as $\text{GCD}(3, \Phi(N)) = 1$

- $3$ is not a factor of $(p-1)$ or $(q-1)$

How does Alice compute $x = y^{1/3} \mod N$?

Security:

- Computing cube roots is hard (assuming RSA)
- Adversary cannot compute $x$
- However, $x$ is distinguishable from a random key
Key Distribution from DH

Everyone agrees on group $G$ or prime order $p$

$a \leftarrow \mathbb{Z}_p$

$b \leftarrow \mathbb{Z}_p$
Key Distribution from DH

Everyone agrees on group $G$ or prime order $p$

$a \leftarrow \mathbb{Z}_p \quad g^a \quad g^b \quad b \leftarrow \mathbb{Z}_p$
Key Distribution from DH

Everyone agrees on group $G$ or prime order $p$

$a \leftarrow \mathbb{Z}_p \quad g^a \quad g^b \quad b \leftarrow \mathbb{Z}_p$

$k = (g^b)^a = g^{ab} \quad k = (g^a)^b = g^{ab}$
Key Distribution from DH

**Theorem:** If DDH holds on $G$, then the Diffie-Hellman protocol is secure

Proof:
- $(\text{Trans}, k) = (((g^a, g^b), g^{ab})$
- DDH means indistinguishable from $((g^a, g^b), g^c)$

What if only CDH holds, but DDH is easy?
Today

Public key encryption
Public Key Encryption
Public Key Encryption
Public Key Encryption

\[ c \leftarrow \text{Enc}(pk, m) \]
Public Key Encryption

\[ c \leftarrow \text{Enc}(pk,m) \]

\[ m \leftarrow \text{Dec}(sk,c) \]
Public Key Encryption

$c \leftarrow \text{Enc}(pk, m)$

$m \leftarrow \text{Dec}(sk, c)$
PKE vs Key Agreement

Key agreement:

$K_{AB}$

$K_{AB}$
PKE vs Key Agreement

Key agreement:

\[ k_{AB} \]
PKE vs Key Agreement

Key agreement:

Alice: $k_{AB}$, $k_{AC}$

Bob: $k_{AB}$

Charlie: $k_{AC}$
PKE vs Key Agreement

Key agreement:

For $n$ users, need $O(n^2)$ key exchanges
PKE vs Key Agreement

PKE:

$$sk_A \rightarrow pk_A$$
PKE vs Key Agreement

PKE:

\[ \text{PKE:} \]

\[ \text{sk}_A \quad \text{pk}_A \quad \text{pk}_B \quad \text{sk}_B \]

[Diagram showing Alice (sk_A), Bob (pk_B), and Charlie (unknown context)]
PKE vs Key Agreement

PKE:

For \( n \) users, need \( O(n) \) public keys
PKE Syntax

Message space $\mathcal{M}$

Algorithms:
• $(sk,pk) \leftarrow \text{Gen}(\lambda)$
• $\text{Enc}(pk,m)$
• $\text{Dec}(sk,m)$

Correctness:
$\Pr[\text{Dec}(sk,\text{Enc}(pk,m)) = m: (sk,pk) \leftarrow \text{Gen}(\lambda)] = 1$
Security

One-way security

Semantic Security

CPA security

CCA Security
One-way Security

$\text{Gen()} \rightarrow (sk, pk)$

$m \leftarrow M$

$c \leftarrow Enc(pk, m)$

$m' \rightarrow$
Semantic Security

\[ (s,k,p,k) \leftarrow \text{Gen}() \]
\[ c \leftarrow \text{Enc}(p,k,m_b) \]
CPA Security

\((\text{sk, pk}) \leftarrow \text{Gen}()\)

\(c \leftarrow \text{Enc}(\text{pk}, m_b)\)
**Theorem:** An encryption scheme \((Gen,Enc,Dec)\) is semantically secure if and only if it is CPA secure.
CCA Security

$(sk, pk) \leftarrow \text{Gen}()$

c $\leftarrow \text{Enc}(pk, m_b)$

$p_k$ →

c

$m$

$m_0, m_1$

$c^*$

$c \neq c^*$

$m$

$b'$
One-way Encryption from RSA

**Gen():**
- Choose random primes $p, q$
- Let $N = pq$
- Choose $e, d$ s.t. $ed = 1 \mod (p-1)(q-1)$
- Output $pk = (N, e)$, $sk = (N, d)$

**Enc(pk, m):** Output $c = m^e \mod N$

**Dec(sk, c):** Output $m' = c^d \mod N$
Theorem: If the RSA one-way function is secure for $e$, then the RSA encryption is one-way secure

Proof: adversary sees exactly output of one-way function and is asked to invert
Considerations

(pk) Alice

(sk) Server

(pk) Minion

(pk) Charlie Brown
Considerations

A single server often has to decrypt many ciphertexts, whereas each user only encrypts a few messages.

Therefore, would like to make decryption fast.
Considerations

Encryption running time:
• $O(\log e)$ multiplications, each taking $O(\log^2 N)$
• Overall $O(\log e \log^2 N)$

Decryption running time:
• $O(\log d \log^2 N)$

(Note that $ed \geq \Phi(N) \approx N$)
Considerations

Possibilities:
• \( e \) tiny (e.g. 3): fast encryption, slow decryption
• \( d \) tiny (e.g. 3): fast decryption, slow encryption
  • Problem?
• \( d \) relatively small (e.g. \( d \approx N^{0.1} \))
  • Turns out, there is an attack that works whenever \( d < N^{0.292} \)

Therefore, need \( d \) to be large, but ok taking \( e=3 \)
Considerations

Chinese remaindering to speed up decryption:

• Let $sk = (d_0, d_1)$ where
  \[ d_0 = d \mod (p-1), \quad d_1 = d \mod (q-1) \]

• Let $c_0 = c \mod p, \quad c_1 = c \mod q$

• Compute $m_0 = c^{d_0} \mod p, \quad m_1 = c^{d_1} \mod q$

• Reconstruct $m$ from $m_0, m_1$

Running time:

• $r \log^3 p + r \log^3 q + O(\log^2 N) \approx r(\log^3 N)/4$
CPA security from RSA?

Use hardcore bit for RSA func:

- \( \text{Enc}(pk,m) : r \leftarrow \mathbb{Z}_N^*, c = (r^e \mod N, h(r) \oplus m) \)

**Theorem:** If RSA is one-way and \( h \) is hardcore for RSA, then this encryption scheme is CPA secure

**Proof:**

\[
(r^e \mod N, h(r) \oplus m_0) \approx (r^e \mod N, b \oplus m_0) \\
\approx (r^e \mod N, b \oplus m_1) \approx (r^e \mod N, h(r) \oplus m_1)
\]
Goldwasser-Micali

**Gen()**:  
- Choose random primes $p,q$  
- Let $N=pq$  
- Choose $x$ a quadratic non-residue mod $p$ and $q$  
- Output $pk=(N,x)$, $sk=(p,q)$

**Enc**(pk,$m\in\{0,1\}$): $r\leftarrow\mathbb{Z}_N^*$, $c\leftarrow x^mr^2 \mod N$  
- If $m=0$, then $c$ is a quadratic residue  
- If $m=1$, then $c$ is a non-residue
Determining Residues

Let $c \in \mathbb{Z}_p^*$ for a prime $p$
How to test if $c$ is a quadratic residue?

Let $c \in \mathbb{Z}_N^*$ for $N=pq$
- If you know $p$ and $q$, test for residuosity mod $p$ and $q$ ⇒ QR mod $N$ iff QR mod both $p$ and $q$
- If you don’t know factors, presumed hard
**Definition:** The Quadratic Residuosity problem mod $N=pq$ is to distinguish a random QR from a random $x$ that is not a QR mod $p$ or mod $q$.

**Theorem:** If the QR problem is hard, then Goldwasser Micali is CPA secure.
Theorem: If the QR problem is hard, then Goldwasser Micali is CPA secure

Proof:
• Hybrid 0: \( pk \) is honestly generated, encrypt \( m_0 \)
• Hybrid 1: \( pk \) is a random QR, encrypt \( m_0 \)
• Hybrid 2: \( pk \) is a random QR, encrypt \( m_1 \)
• Hybrid 3: \( pk \) is honestly generated, encrypt \( m_1 \)
Bit encryption $\Rightarrow$ Multi-bit

Let $\text{Gen, Enc, Dec}$ be an encryption scheme for bits

$\text{Gen}'() = \text{Gen}()$

$\text{Enc}'(pk, (m_1, ..., m_n)) = (\text{Enc}(pk, m_1), ..., \text{Enc}(pk, m_n))$

$\text{Dec}'(sk, (c_1, ..., c_n)) = (\text{Dec}(sk, c_1), ..., \text{Dec}(sk, c_n))$

**Theorem:** If $(\text{Gen, Enc, Dec})$ is CPA secure, then so is $(\text{Gen}', \text{Enc}', \text{Dec}')$
ElGamal

Group $G$ of order $p$, generator $g$
Message space = $G$

$Gen()$: 
• Choose random $a \leftarrow \mathbb{Z}_p^*$, let $h \leftarrow g^a$
• $pk=h$, $sk=a$

$Enc(pk,m \in \{0,1\})$: 
• $r \leftarrow \mathbb{Z}_p$
• $c = (g^r, h^r \times m)$

$Dec?$
**Theorem:** If DDH is hard in $G$, then ElGamal is CPA secure

Proof:

- Adversary sees $h = g^a, g^r, g^{ar} \times m_0$
- DDH: indistinguishable from $g^a, g^r, g^c \times m_0$
- Same as $g^a, g^r, g^c \times m_1$
- DDH again: indistinguishable from $g^a, g^r, g^{ar} \times m_0$
PKE from One-Round Key Exchange
PKE from One-Round Key Exchange

Here, $state_A$, $state_B$, are the internal states of $A,B$ after first message
PKE from One-Round Key Exchange

Gen(): Run A(), getting x, and state_A
• sk = (x,state_A), pk = x

Enc(pk,m):
• Run B(x) to get y and state_B,
• Run B(state_B, x) to get k
• c = (y, k⊕m)

Dec(sk, (y,d) ):
• Run A(state_A, x, y) to get k
• m← d⊕k
PKE from One-Round Key Exchange

**Theorem:** If \((A,B)\) is a secure one-round key exchange protocol, then \((\text{Gen,Enc,Dec})\) is CPA secure.

Proof:

\((pk, c) = (x, y, d)\) is exactly what the adversary would see if:
- Run key agreement protocol to get \(k\)
- Encrypt \(m\) using \(k\) as OTP
One-Round Key Exchange from PKE

\[ pk \]
\[ c = \text{Enc}(pk, k) \]
\[ k \leftarrow \text{Dec}(sk, c) \]
\[ k \leftarrow M \]
Black Box Separations

Recall: hard to build key agreement from one-way functions

Therefore, also hard to build PKE from one-way functions

Appears we must rely on number theory for PKE
Practical Considerations

Number theory is computationally expensive
  • Need big number arithmetic

Symmetric crypto (e.g. block ciphers) much faster

Want to minimize use of number theory, and rely mostly on symmetric crypto
Hybrid Encryption

Let \((\text{Gen}_{\text{PKE}}, \text{Enc}_{\text{PKE}}, \text{Dec}_{\text{PKE}})\) be a PKE scheme, 
\((\text{Enc}_{\text{SKE}}, \text{Dec}_{\text{SKE}})\) a SKE scheme

\[
\begin{align*}
\text{Gen}() &= \text{Gen}_{\text{PKE}}() \\
\text{Enc}(pk, m) &: k \leftarrow K, c = (\text{Enc}_{\text{PKE}}(pk, k), \text{Enc}_{\text{SKE}}(k, m)) \\
\text{Dec}(sk, (c_0, c_1)) : \\
& \cdot k \leftarrow \text{Dec}_{\text{PKE}}(sk, c_0) \\
& \cdot m \leftarrow \text{Dec}_{\text{SKE}}(k, c_1)
\end{align*}
\]

Now PKE used to encrypt something small (e.g. 128 bits), SKE used to encrypt actual message (say, GB’s)
Hybrid Encryption

**Theorem:** If \( (\text{Gen}_{\text{PKE}}, \text{Enc}_{\text{PKE}}, \text{Dec}_{\text{PKE}}) \) is CPA secure and \( (\text{Enc}_{\text{SKE}}, \text{Dec}_{\text{SKE}}) \) is one-time secure, then \( (\text{Gen,Enc,Dec}) \) is CPA secure

Hybrid 0: \( (\text{Enc}_{\text{PKE}}(pk,k), \text{Enc}_{\text{SKE}}(k,m_0)) \)
Hybrid 1: \( (\text{Enc}_{\text{PKE}}(pk,k'), \text{Enc}_{\text{SKE}}(k,m_0)) \)
Hybrid 2: \( (\text{Enc}_{\text{PKE}}(pk,k'), \text{Enc}_{\text{SKE}}(k,m_1)) \)
Hybrid 3: \( (\text{Enc}_{\text{PKE}}(pk,k), \text{Enc}_{\text{SKE}}(k,m_1)) \)
Next Time

Trapdoor Permutations

Begin: digital signatures (aka public key MACs)