Lecture 11:
Public Key Encryption, definitions and case studies

Diffie-Hellman '76: New Directions in Cryptography
```
```
```
```

```
```
```

```
```
```

```
```
```

```
```
```

```
```
```

```
```
```

```
```
```

```
```
```

```
```
```

```
```
```

```
```
```

New Ideas: Public Key Encryption

Digital Signatures

New Goal: Base Security on Computational Complexity: Design systems provably hard to break for a computationally bounded adversary

Public Key Encryption [Ref1]

- Users A, B, C ... in a network have each a pair of keys <Public-Key, Secret-key>
- Every user generates his pair of keys <P_u, S_u> using a key generation algorithm G, publishes his public key P_u in a public directory, and keeps secret S_u.
- To send a secret message m to user U, user B looks up P_u and sends c = E(P_u, m) to U where E is an encryption algorithm
- Having received c, U computes D(S_u, c) to recover m where D is a decryption algorithm
- Clearly for this to work D(S_u, E(P_u, m)) = m

Let k be the security parameter.

<table>
<thead>
<tr>
<th>Users</th>
<th>Public Directory</th>
<th>Secret Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>P_A</td>
<td>S_A</td>
</tr>
<tr>
<td>B</td>
<td>P_B</td>
<td>S_B</td>
</tr>
<tr>
<td>C</td>
<td>P_C</td>
<td>S_C</td>
</tr>
</tbody>
</table>

A particular PKC is specified by a triplet (G, E, D) of probabilistic polynomial time algorithms known to all (assume all keys of length <k)
Simple Remarks to check understanding

- The encryption and decryption functions are publicly known and are easy to compute if all the inputs are known.
- It is possible to generate pairs of keys $P_u, S_u$ such that $D(S_u, E(P_u, m)) = m$.
- It is hard to decrypt knowing only $P_u$, and in particular hard to compute $S_u$.
- Everybody can send messages to $U$ without need of meeting $U$ in advance.
- The service maintaining the public file cannot decrypt, it is only trusted to publish the right key for the right user.

Public Key Encryption (formal)

Definition: A public-key encryption scheme is a triple $(G, E, D)$ of PPT algorithms s.t.

- **Key Generation**: $G$ takes as input the security parameter $k$, and produces a pair of keys $(e, d)$.

- **Encryption**: $E$ takes as input public key $e$ from range of $G$ and cleartext message $m$ and outputs ciphertext $c$.

- **Decryption**: $D$ takes as input secret key $d$ from range of $G$ and ciphertext $c$ from range of $E(e, m)$ and outputs $m'$ s.t. $m = m'$ (i.e., $D(d, E(e, m)) = m$).

System is Secure: Leave Vague for now.

Comments on the Definition

- At this point no difference between definition of public key encryption or private key encryption (difference in security requirement)
- For simplicity assume that messages are length $k$ (the security parameter)
- Both $E$, $D$ are probabilistic which will allow greater flexibility and possibly greater security

First PKC: trapdoor function model

Let $M$ be a message space, messages are length $k$. Diffie and Hellman suggest the following $(G, E, D)$ as a PKC, given the collection of trapdoor functions $F$.

- **Key Generation**: $G(k)$ outputs pair $(f, t)$ where $f \in F$ and $t$ is the associated trapdoor information.

- **Encryption**: $E(f, m) = f(m)$.

- **Decryption**: $D(t, c) = D(t, f(m)) = f^{-1}(c) = f^{-1}(f(m)) = m$.

Note: $(G, E, D)$ exist by properties (generation, evaluation, and trapdoorness) of $F$. 
Recall: Collections of Trapdoor Functions

Definition: A collection of trapdoor functions $F = \{f_i : D_i \rightarrow D_i\}_{i \in I}$ is a collection of one-way functions s.t.

- **Generation:** \( \exists \) PPT algorithm \( G \) that on input security parameter \( k \) selects a random \( f_i \) in \( F \) along with short trapdoor information \( t_i \) (where \( |i| = k \)).
- **Trapdoor-ness:** \( \exists \) PPT algorithm \( I \), s.t. \( I(f_i(x), t_i) = x' \) such that \( f_i(x) = f_i(x') \)

Public Key Encryption Based on Trapdoor Functions

How Secure is this beautiful idea

- Clearly as \( F \) is a collection of trapdoor functions then by the Hard-to-Invert property (without the trapdoor) of trapdoor functions, it is computationally impossible to invert \( f(x) \) for a random \( x \).
- Is that enough? Let’s talk about security ...

Problems with Security (1)

- **Special Message Spaces:** The fact that \( f \) is a trapdoor function does not mean that inverting \( f(x) \) is hard when \( x \) is from a special message space, e.g. \( M = \{0,1\} \).
- **Partial Information:** The fact that \( f \) is a trapdoor function does not imply \( f(x) \) hides all partial information about \( x \), e.g. \( \text{lsb}(x) \) leaks from \( g^x \mod p \), Jacobi Symbol leaks from \( x^e \mod n \). In fact, for any OWF \( f \) some information leaks about \( x \) from \( f(x) \). Moreover, any deterministic \( E \) leaks partial information.
Problems: \( f(x) \neq x \)

Relationships between Cipher texts: Can detect relations between \( f(x) \) for related \( x \), e.g. for RSA with exponent \( e \), if the same message is sent encrypted to \( e \) different users it can be recovered.

What do we actually want?

MAYBE WE ARE ASKING TOO MUCH?
WE WILL SEE THAT CAN ACHIEVE IT

The RSA Public Key Cryptosystem 77

Invented by Rivest-Shamir-Adleman in 1977

\[ \text{RSA} = \{ \text{RSA}_{n,e} : \mathbb{Z}_n^* \rightarrow \mathbb{Z}_n^*, \]
\[ \text{RSA}_{n,e}(x) = x^e \mod n \} \]

Key Generation: User A chooses random primes \( p, q \), and \( e \) s.t. \( \gcd(e, \phi(n)) = 1 \), sets \( n = pq \), computes \( d \) s.t. \( ed = 1 \mod \phi(n) \), and publishes \((n,e)\) in public directory

\[ E((n,e),m): \text{To send to user A, look up } (n,e) \text{ and compute } c = m^e \mod n \]

\[ D(d,c): \text{User A computes } c^d \mod n = m^{ed} \mod n = m \]

Recall, RSA and Factoring Integers

- **Fact 1:** Given \( n, e, p, \) and \( q \), its easy to compute \( \phi(n) \) and \( d = e^{-1} \mod \phi(n) \).
- **Fact 2:** Given only \( n,e \), computing \( \phi(n) \) is as hard as factoring \( n \)
- **Fact 3:** Given only \( n,e \), computing \( d \) is as hard as factoring \( n \)

- **Conclusions:**
  - If can factor, can invert RSA
  - But, is Inverting (breaking) RSA as hard as factoring? **MAJOR OPEN PROBLEM**

Running Times

- **Generation** \( O(k^4) \)
- **Encryption** \( O(k^3) \)
- **Decryption** \( O(k^3) \)
  with trapdoor

Efficiency is implementation specific (software, hardware)
Security of RSA as PKC

Cannot use for small message spaces

Partial Information: RSA(x) leaks Jacobi Symbol (x/n), mental poker example.

Low Exponent Attacks 1: Say e=3 (popular choice)
Suppose that two cipher texts are sent \( c_1 = m^3 \mod n \) and \( c_2 = (m+1)^3 \mod n \). Can now solve for \( m = \frac{(c_2 + 2c_2 - 1)}{(c_2 - c_1 + 2)} \).

This attack can be generalized to messages \( m \) and \( am+b \), \( e > 3 \), and for \( k \) messages related via higher degree polynomial.

- Low Exponent Attacks 2: if send same encrypted \( m \) to \( e \) different users using same \( e \), can decrypt

Bit Security of RSA Function

Recent Result: for all \( i \), computing the \( i \)-th bit of \( x \) from RSA \( n,e(x) \) with probability better than \( 1/2 + \) non-neg(k) is as hard as as inverting RSA itself (\(|n|=k\)).

Reduction-Theorem: If \( \exists \) PPT algorithm \( A \) s.t.
\( \text{Prob}[A(n,e, RSA_{n,e}(x))=x_i] > 1/2 + \) non-neg(k) (over \( x \) in \( Z_n^* \)),
then \( \exists \) PPT algorithm for inverting RSA_{n,e}.
Conclusion: RSA hides \( i \)-th bit extremely well

So, is all partial information hidden about RSA(x)?

- No, there exist bits of information about \( x \) that can be leaked from RSA_{n,e}(x) --- \( \frac{x}{n} \) Jacobi symbol

- Who cares? Actually, means a bug in a mental poker protocol of SRA, pointed out by Lipton.
**Textbook RSA/Rabin is insecure**

Textbook RSA encryption is an insecure cryptosystem:
- Does not satisfy basic definitions of security.
- Many attacks exist.

- The RSA trapdoor permutation can be used as a building block!

**RSA encryption in practice**

- Never use textbook RSA.
- RSA in practice:
  - Main question:
    - How should the preprocessing be done?
    - Can we argue about security of resulting system?

**PKCS1 V2.0 - OAEP**

- New preprocessing function: OAEP (BR94).

- **Theorem**: when \( H,G \) are "random oracles" scheme is secure

- In practice: use SHA-1 or MD5 for \( W,G,H \)

**Random Oracle Model Methodology**

[FS86, BR93]

1. Design ideal cryptographic system in ROM
2. Prove ideal system is secure
3. Replace oracle by a function \( h \) which has a "real life" implementation.

HOPE: ROM methodology produces secure and efficient crypto systems.
Random Oracle Widely Used in Practical Design of Cryptosystem

- RSA-OAEP encryption scheme is in
  - RSA PKCS #1 v2.0
  - IEEE P1363
  - Implemented in products: SET, CDSA
- PSS probabilistic signature scheme is in
  - IEEE P1363a draft standard
- Both schemes use random oracle to boost security of underlying scheme

What Does ROM security mean in the real world?

Beware:
Possibly there exists no implementation of h in the real world that could make the schemes secure??

\[ x \xrightarrow{h(x)} h \]

Random Oracle (ROM)

[CGH98]: \exists\text{ encryption scheme (and signature scheme) secure in ROM and insecure w.r.t. any implementation of } h \text{ (diagonalization)}

Implementation attacks

- Attack the implementation of RSA.
- Timing attack: (Kocher 97)
  The time it takes to compute \( C^d \mod N \) can expose \( d \).
- Power attack: (Kocher 99)
  The power consumption of a smartcard while it is computing \( C^d \mod N \) can expose \( d \).
- Faults attack: (BDL 97)
  A computer error during \( C^d \mod N \) can expose \( d \).

RSA in the “Real World”

- Part of many standards: PKCS, ITU X.509, ANSI X9.31, IEEE P1363
- Used by: SSL, PEM, PGP, Entrust, ...

- The standards specify many details on the implementation, e.g.
  - \( e \) should be selected to be small, but not too small
  - “multi prime” versions make use of \( n = pqr \)...
    this makes it cheaper to decode especially in parallel (uses Chinese remainder theorem).
Computing Square Roots mod Composites As Hard As Factoring

Theorem:
If ∃ PPT algorithm A s.t. A(n,x^2 \mod n)=x, then ∃ PPT algorithm to factor n.

Proof: On input n, choose a random r in Z_n^*. Compute x=A(n,r^2 \mod n).
Now x^2 = r^2 \mod n, and
x \neq \pm r \mod n (with prob 1/2).
x^2 = r^2 \mod n \rightarrow (x-r)(x+r) = 0 \mod n
either p| (x-r) or q | (x-r),
gcd(x-r,n) = p or q    QED

Factoring Based Public Key Cryptosystem

Use the trapdoor collection of
Rabin/BW = \{ BW_n: QR_n \Rightarrow QR_n
\}
BW_n(x) = x^2 \mod n\}_{n=pq}

Key Generation: User A chooses random primes p,q such that p=q=3 \mod 4, publishes n=pq in public directory, and keeps p,q secret
E(n,m): send c = m^2 \mod n
D({p,q},c): compute the square root of c which is itself a square.

Efficiency: O(k^2) encryption, O(k^3) decryption
Similar Properties/Problems to RSA
Special Problem: Completely breakable under chosen cipher text attack

More PKC based on Trapdoor Function Model

Instead of RSA collection, use the trapdoor functions collection equivalent to factoring.
Let n=pq, p,q primes where p=q=3 \mod 4 and recall BW_n(x)=x^2 \mod n, where BW_n: QR_n \Rightarrow QR_n

• Public Key: n,
• Secret Key: factors p,q of n
• To encrypt m: E(m) = m^2 \mod n
  To decrypt m: D(c) = computes square root which is itself a square.

Efficiency: O(k^2) encryption, O(k^3) decryption
Similar Properties/Problems to RSA
Special Problem: Completely breakable under chosen cipher text attack

How to Build a PKC which is provably secure by our computationally indistinguishable under Factoring/RSA assumptions?

More generally, any trapdoor function
Brainstorming: What Should Security Mean?

- Should not be able to recover Secret Key
- Should not be able to recover message for any message space, e.g. english, turkish, \{0,1\}
- Should not be able to compute partial information about messages
- Should not be able to compute useful information from traffic of messages
- Should be safe to use encryption as a building block in protocol design

With high Probability

Computational Indistinguishability[GM]

A scheme is secure if no adversary can distinguish between $E(m_0)$ and $E(m_1)$ better than 50-50, for any two messages $m_0$, $m_1$ chosen by the adversary.

Clearly, no deterministic scheme can be secure.

Computational Indistinguishability (Formal)

**Definition:** Say that a PKC $(G,E,D)$ is computationally indistinguishable if $\forall$ PPT algorithms $M,A$ s.t. $\Pr[A(e,m_0,m_1,c)=m] < 1/2 + \text{neg}(k)$

(where $(e,d) \in G(k)$, $(m_0,m_1) \in M(k)$, $m \in \{m_0,m_1\}$, $c \in E(e,m)$)

**Remarks:**
- Would be stronger to say for all $m_0,m_1$, but do not know how to prove it, unless messages are independent of public keys
- In private-key encryption, $A$ does not get $e$
- Note that even if adversary knows that the message encrypted is one of two messages (which he himself chose), still cannot tell them apart

Semantic Security[GM]

An alternative definition proposed is similar to Shannon’s definition of security (adapted to bounded adversary)

**Definition:** A PKC $(G,E,D)$ is semantically secure if for all message spaces $M$, and for all information functions $f: M \rightarrow V$, for all PPT algorithms $A$,

$| \Pr[\text{A wins game 1}] - \Pr[\text{A wins game 2}] | < \text{neg}(k)$

where

**Game 1:** Choose $m$ in $M(k)$. Ask adversary $A$ to compute $h(m)$

**Game 2:** Choose $m \in M(k)$, give $A$ cipher text $c \in E(e,m)$

where $(e,d) \in G(k)$. Ask $A$ to compute $h(m)$. 
How to Build a PKC satisfying Above Definitions

Abandon trapdoor function PKC model in favor of probabilistic Encryption algorithms

Key first step:
How to securely encrypt single bit messages?

Two Paradigms: How to Send a Single Bit Securely

• Hiding Bits in One Way and Trapdoor Functions
• Trapdoor Predicates (or hard decision problems)

First Idea

Probabilistic Encryption of Single Bits: Trapdoor Function Model \cite{GM,Y82}

Let f be a trapdoor function, B hard-core predicate

Listener can’t distinguish

To send m=0 send a random \( f(x) \) s.t. \( B(x)=0 \)
To send m=1 send a random \( f(x) \) s.t. \( B(x)=1 \)
Probabilistic Encryption: Trapdoor Function Model

Listener can’t distinguish
To send 0 pick \( f(x) \) at random
To send 1 pick \( f(x) \) at random
To send \( m=0010... \)
Pick \( f(x_1) \ f(x_2) \ f(x_3) \ f(x_4) \) s.t. \( B(x_i) = m_i \)

Probabilistic Public-Key Encryption

Let \( F \) be a collection of trapdoor functions with hard core bit \( B \), define probabilistic public-key encryption \( \text{PE}_F = (G,E,D) \):

- \( G(k) \) chooses \( f \in F \) with trapdoor \( t \)
- \( E(f,m=m_1...m_k) \): For \( k \geq j \geq 1 \), choose \( x_j \) s.t. \( B(x)_j = m_j \). Output \( c = (f(x_1)f(x_2)...f(x_k)) \)
- \( D(t,c) \): For \( 1 < j < k \), compute \( m_j = f^{-1}(f(x_j)) \)

Theorem: if \( F \) is a collection of trapdoor functions, then probabilistic encryption \( \text{PE}_F \) is computationally indistinguishable (in particular, if and only if factoring integers is hard)
Performance Comparison of \( \text{PE}_{\text{RSA}} \) to RSA

Let \(|m| = k\):

- **Size of Public key**: \(O(k)\) for both
- **Bandwidth**: RSA \(O(k)\), PE \(O(k^2)\)
- **Computation cost**: RSA \(O(k^3)\), PE \(O(k^4)\)

**Conclusion**: high price to pay for security

Wrong Conclusion: think again

---

Efficient Probabilistic Public Key Encryption (EPEF) \([BG]\)

- **Key Generation**: Choose \(p, q\) primes s.t. \(p = q = 3 \mod 4\) and set \(n = pq\). Public key is \(n\), and secret key is the factorization
- **\(E(n, m)\)**: On input message \(m\), First, choose \(r\) in \(\mathbb{Z}_n^*\) and compute
  \[
  \text{pad} = \text{lsb}(r)\text{lsb}(r^2 \mod n)\text{lsb}(r^4 \mod n) \ldots \text{lsb}(r^{2^{k+1}} \mod n)
  \]
  Set \(c = (\text{pad} \oplus m, r^{2^{k+1}} \mod n)\)
- **\(D((p,q),c)\)**: On \(c = (u,v)\), compute \(r = k^{th}\) root of \(v\) mod \(n\) (using factorization), re-compute \(\text{pad}\), and set \(m = u + \text{pad}\).

---

Performance comparison of \( \text{PE}_{\text{RABIN}} \) and RSA

Let \(|m| = k\):

- **Size of Public key**: \(O(k)\) for both
- **Bandwidth**: RSA \(O(k)\), PE \(O(2k)\)
- **Computation cost**: RSA \(O(k^3)\), PE \(O(k^3)\)
- **Security**: EPE Major Win!!

Performance comparison of \( \text{PE}_{\text{RABIN}} \) and RSA(2)

Let \(|m| = S\):

- **Size of Public key**: \(O(k)\) for both
- **Bandwidth**: RSA \(O(S+k)\), PE \(O(S+k)\)
- **Computation cost**: RSA \(O(k^3)\), PE \(O(k^3)\)
- **Security**: EPE Major Win!!
In general, the EPE idea essentially is:

Let \( E(m) = (f(R), G(R) + m) \)

for \( R \) random, \( f \) trapdoor permutation,
\( H(R) \) pseudo random sequence with seed \( R \) is secure. (oracle hash \([BR,CM]\)).

Second Idea

- Actually, Trapdoor functions may be stronger than what’s required to achieve public key encryption which is secure.

Trapdoor Predicates imply Probabilistic Public-Key Encryption

In fact, trapdoor predicates may suffice.

Trapdoor Predicates \( B(x) \to \{0,1\}: \)

- Easy to sample in \( B(x)=0, B(x)=1 \)
- Hard to guess \( B(y) \) better than 50-50
- With trapdoor, Easy to compute \( B(y) \)

To Encrypt \( b \): Send random \( x \) s.t \( B(x) = b \).

Probabilistic Encryption Based on Trapdoor Predicates (Square/Non-Square mod Composites Problem)

- **key Generation**: Choose \( p,q \) primes where \( p=q=3 \mod 4 \). Let \( n=pq \). Set public key = \( n \), and secret key={\( p,q \)}
- **E(\( n,m \))**: Suppose \( m \) is a single bit. Choose \( x \) in \( \mathbb{Z}_n^* \) at random and let \( y=x^2 \mod n \).
  - If \( m=0 \), send \( y \) (a square mod \( n \))
  - else send -\( y \) (a non-square mod \( n \) with Jacobi symbol +1)
- **D(\( (p,q),c \))**: Test if \( y \) is a square mod \( p \) and mod \( q \).
  - If yes, \( m=0 \), else \( m=1 \).

**Claim**: Scheme is Computationally indistinguishable if and only if QRA assumption.
**Diffie Hellman Key Exchange 76**  
*Discrete Log Based*

Two parties A and B share a prime, p, and generator, g.

**Party A** chooses 1 < x < p-1 at random, set y = g^x mod p, and sends y to B over public channel.

**Party B** chooses 1 < z < p-1 at random, set w = g^z mod p, and sends w to A over public channel.

**Joint Secret Key** of A and B = g^{xz} mod p =

- A can compute: y^z mod p
- B can compute: w^x mod p

Now can use any private key system they want.

**Remarks:** Security based on DHP.  
First key Exchange over public channels proposed!

---

**Key Exchange Requirements**

Allows two users who have never met and do not have any a-priori common information.

To decide on a **common key**:

- known to them
- unknown to anyone else who listened to all of their communication

---

**El Gamal Cryptosystem 84**  
*Discrete Log Based*

Recall EXP_{p,g}(x) = g^x mod p

**key Generation:** choose p, g, 1 < x < p-1 at random, set y = g^x mod p, publish (p, g, y) and keep x secret.

**E((p,g,y),m):** choose random 1 < r < p-1, set pad = (y)^r mod p = g^{rx} mod p, and send c = (pad * m, g^r).

**D(x,c):** On c = (u, v), compute pad = (v)^x = g^{rx} mod p and set m = u/pad.

**Remarks:** g, p can be shared by all users, security is based on DHP hardness.  
Scheme is probabilistic, E tosses coins!!