Introduction to Modern Cryptography

Benny Chor

RSA: Review and Properties
Factoring Algorithms
Trapdoor One Way Functions
PKC Based on Discrete Logs (Elgamal)
Signature Schemes

Lecture 8

Tel-Aviv University

4 March 2008 (slightly revised 22 March)
Square Roots of $1 \quad Z_{pq}^*$ (reminder)

- in $Z_p^*$, $1$ has two square roots: $1$ and $p - 1$.
- in $Z_q^*$, $1$ has two square roots: $1$ and $q - 1$.
- What about the square roots of $1$ in $Z_{pq}^*$?

- $y^2 = 1 \pmod{pq}$ iff $y^2 = 1 \pmod{p}$ and $y^2 = 1 \pmod{q}$.
- So $y = \pm 1 \pmod{p}$ and $y = \pm 1 \pmod{q}$.
- This gives rise to four systems of modular equations
  1. $y = 1 \pmod{p}$ and $y = 1 \pmod{q}$.
  2. $y = -1 \pmod{p}$ and $y = -1 \pmod{q}$.
  3. $y = 1 \pmod{p}$ and $y = -1 \pmod{q}$.
  4. $y = -1 \pmod{p}$ and $y = 1 \pmod{q}$.
- The solution to (1) is $y_1 = 1$.
- The solution to (2) is $y_2 = pq - 1 = -1 \pmod{p}$.
- The solutions to (3) and (4) are obtained using the Chinese remainder theorem. Denote them by $y_3$ and $y_4 = pq - y_3 = -y_3 \pmod{p}$.
General Square Roots in $\mathbb{Z}_{pq}^*$

In general, the square roots of $z^2$ are any of the four square roots of $1 \pmod{pq}$, multiplied by $z$. Specifically, the four square roots of $z^2$ in $\mathbb{Z}_{pq}^*$ are

1. $z_1 = z$,
2. $z_2 = -z = pq - z$,
3. $z_3 = y_3 \cdot z \pmod{pq}$,
4. $z_4 = y_4 \cdot z = -z_3 \pmod{pq}$.

- It is not hard to see that more than four square roots of any of $z^2$ in $\mathbb{Z}_{pq}^*$ imply more than two square roots in $\mathbb{Z}_p^*$ or in $\mathbb{Z}_q^*$, a contradiction.
- Thus every square in $\mathbb{Z}_{pq}^*$ has exactly four square roots.
- The mapping $x \mapsto x^2 \mod pq$ is a four to one mapping.
- So the number of quadratic residues in $\mathbb{Z}_{pq}^*$ is $(p - 1)(q - 1)/4$. 
The RSA Public Key Cryptosystem (reminder)

- Bob’s private information: two large primes $p, q$.
- Public information: Their product, $m = p \cdot q$. An integer $e$ that is relatively prime to $\phi(m) = (p - 1) \cdot (q - 1)$.
- More private information: An integer $d$ that is relatively prime to $\phi(m) = (p - 1) \cdot (q - 1)$ and satisfies $d \cdot e = 1 \mod \phi(m)$.
- Messages $P$ are elements in $\mathbb{Z}_m$, namely numbers in $[1, \ldots, m - 1]$. Almost surely they are relatively prime to $m$.
- To encrypt $P$, compute $C = P^e \pmod{m}$, and send $C$ to Bob.
- To decrypt $C$, Bob computes $C^d = P^{d \cdot e} = P \pmod{m}$.
RSA and Factoring

- If Eve could factor $pq$, she obtains all private information of Bob.
- Hence breaking RSA cannot be harder than factoring.
- How hard is it to compute the secret key, $d$, from the public information $m, e$?
- Recall $d \cdot e = 1 \pmod{\phi(m)}$, so $d \cdot e - 1 = C \cdot (p - 1)(q - 1)$ for some unknown, positive integer $C$.
- If Eve can find $d$, then she can easily compute $h = ed - 1 = C(p - 1)(q - 1)$.
- Note that computing $(p - 1)(q - 1)$ from $h = C(p - 1)(q - 1)$ seems to require factoring.
RSA and Factoring (2)

- Eve holds $h = C(p - 1)(q - 1)$, but not $C$ or $(p - 1)(q - 1)$.
- Miller has shown that under ERH, such multiple $C(p - 1)(q - 1)$ enables factoring $m$.
- If we are willing to use randomization (à la Rabin), we can do without the ERH.
- This may sound mysterious, but in fact you already know this (even though you do not know that you know, etc.).
- Let $h = 2^k \cdot r$, with $r$ odd.
- Since $h$ is a multiple of $(p - 1)(q - 1)$, for any $b \in \mathbb{Z}_{pq}^*$ we have $b^h = 1 \pmod{pq}$. 
RSA and Factoring (3)

- Let $h = 2^k \cdot r$, with $r$ odd.
- Since $h$ is a multiple of $(p - 1)(q - 1)$, for any $b \in \mathbb{Z}_{pq}^*$ we have $b^h = 1 \pmod{pq}$.
- We compute the $k + 1$ powers of $b$, $b^{h/2^k}, b^{h/2^{k-1}}, \ldots, b^h$, in $\mathbb{Z}_{pq}^*$.
- If for some $i$, $b^{h/2^i} \neq \pm 1$ but $b^{h/2^{i-1}} = 1$, then $b^{h/2^i}$ is a square root of 1 that is not $\pm 1$.
- In this case $\gcd(b^{h/2^i} \pm 1, pq)$ gives either $p$ or $q$.
- Such small $b$ can be found under ERH.
- Alternatively, there are many such $b$’s, so can find one if picking at random.
- Note that this does not imply that “breaking” RSA is equivalent to factoring – just that recovering $d$ is equivalent to factoring.
- See a small example in Maple (next slide).
RSA and Factoring – Maple example

> p := 2^40 - 87; # p is a 40 bit long prime
  
p := 1099511627689

> q := 2^45 - 55; # q is a 45 bit long prime
> m := p * q; phi := (p-1) * (q-1);
  
m := 38685626224546620079346353
  
  \[ \phi := 38685626224510336195629888 \]

> gcd(17, phi);

  
  \[ 1 \]

> e := 17; # e will be the RSA encryption exponent
  
  \[ e := 17 \]

> d := 1/e mod phi; # d is the RSA encryption exponent
  
  \[ d := 13653750432180118657281137 \]

> h := e * d - 1; # h is a multiple of (p-1)(q-1) by an unknown term. If d is known, we can compute h. We now show HOW TO FACTOR m using h.
  
  \[ h := 232113757347062017173779328 \]

> r := h / 2^7;

  
  \[ r := 1813388729273922009170151 \]

> b := 4;

  
  \[ b := 4 \]

> a7 := b ^ (h / 2^7) mod m; m - a7;

  
  \[ a7 := 31328411123579318434082545 \]
  
  \[ 7357215100967301645263808 \]

> a6 := b ^ (h / 2^6) mod m; # a7<>-1 is a square root of 1

  
  \[ a6 := 1 \]

> gcd(p * q, a7 - 1); # so a7 yields the factors of p * q

  
  \[ 35184372088777 \]

> gcd(p * q, a7 + 1);

  
  \[ 1099511627689 \]
Properties (and Weaknesses) of “Textbook RSA”

- Deterministic encryption, thus easy to identify repetitions (like ECB mode encryption).
- RSA is multiplicative: $E(P_1 \cdot P_2) = E(P_1) \cdot E(P_2)$.
- Thus RSA encryption is not a pseudo random function on $\mathbb{Z}_{pq}$.
- It also implies vulnerability to chosen ciphertext attacks (whether or not this is a real threat can be debated).
Random Padding (aka "Salting") of RSA

- Padding the message by a block of random bits: Suppose the length of $pq$ is $n$ bits. Use $\ell$ bits for the message $P$, concatenate with $n - \ell$ random bits string, $r$: $E(r \circ P) = (r \circ P)^e \pmod{pq}$.

- Padding reduces the information rate, but increases security. It can be shown that if $n - \ell$ is very large, then padded RSA is resistant to chosen plaintext attack.

- Of course for security to hold, pad must be random. Choosing $r = \text{hello world}$, or any other fixed text, is not a good practice.

- For protection against chosen ciphertext attack, a combination of fixed and random padding was proposed by RSA labs: Let $P$ be a $\ell$ bit long message. Pad and encrypt by $(00000000 \circ 00000010 \circ r \circ 00000000 \circ P)^e \pmod{pq}$.

- Fixed parts of pad intended to foil multiplication attacks.

- Unfortunately, some chosen ciphertext attacks were later found. Still, scheme is being used.
Real World Usage of RSA

(1) Key exchange.

(2) Digital signatures.
RSA as a One Way Trapdoor Function

Easy: \( x \rightarrow x^e = y \pmod{pq} \) (\( e \) is known).

Hard: \( y \rightarrow y^d = x \pmod{pq} \) (\( d \) is unknown).

Easy with trapdoor information: \( y \rightarrow y^d = x \pmod{pq} \) (when \( d \) is known).
Informal Definition: $f : D \rightarrow R$ is a trap-door one way function if there is a trap-door $s$ such that:

- Without knowledge of $s$, the function $f$ is a one way function.
- Given $s$, inverting the function $f$ is easy.

Example: the function $f_{g,p}(x) = g^x \pmod{p}$ is not a trap-door one way function.

Example: RSA is a trap-door one way function.
General Remark on Public Key Cryptosystems

- PKCs are order of magnitude slower than private key systems. Hence used mainly to exchange keys or signing.
- Under suitable complexity assumptions, PKC are secure, provided we can trust the association of keys with users.
- If I were tricked to send a message using what I think is the public key of Esau, but Jacob (a well known trickster) is the one that can decipher it, then I may be in trouble.

Isaac rejecting Esau, by Giotto di Bondone, 13-14th centuries, Assissi, Italy.

- To achieve secure communication without prior physical contact, have to establish (and trust) centers for distributing certificates.
- Will be discussed (soon) under “public key infrastructure”.

Introduction to Modern Cryptography  Benny Chor
RSA: Review and Properties Factoring Algorithms Trapdoor One Way Functions PKC Based on Discrete Logs (Elgamal) Signature Schemes
Factoring Algorithms

What is the running time (worst case) of factoring algorithms? Let $m$ be an $n$ bits composite. Hardest numbers to factor are the product of two distinct prime numbers $m = pq$, where both $p − 1$ and $q − 1$ have a large prime factor.

A (very partial) list of algorithms:

- Trial division: $O(2^{n/2})$.
- J.M. Pollard’s rho method: $O(2^{n/4})$.
- Quadratic sieve algorithm: $O(e^{(n \log n)^{1/2}})$.
- General number sieve algorithm: $O(e^{(7n)^{1/3} \cdot \log^2 n})$.
- GNS was introduced by J.M. Pollard in 1988, and later refined by many well-known players of the computational number theory community.
Factoring Algorithms

- The general number sieve algorithm is considered the fastest of all published, “general purpose” factoring algorithms. It was employed to factor RSA-200, a 663-bit number (200 decimal digits), on May 2005. The algorithm was implemented on a cluster of 80 2.2 GHz Opterons. Execution took three months.

  RSA-200 =
  2799783391122132787082946763872260162107044678695542853756000992932612840010760934567105295
  5360856061822351910951365788637105954482006576775098580557613579098734950144178863
  178946295187237869221823983
  Factors =
  3532461934402770121272604978198464368671197400197625023649303468776121253679423200058547956528088349
  and
  7925869954478333033347085841480059687737975857364219960734330341455767872818152135381409304740185467

- We will embark upon a much more modest task: Explain Pollard’s rho method (on the board), implement it on a 2.2 GHz Core 2 Duo MacBook, using Maple, and run it to factor an 85-bit number (in approximately two minutes).
Elgamal Public Key Cryptosystem

• We are now going to describe a second PKC, designed by Taher Elgamal in 1985 (when he was with Netscape).

• Elgamal PKC is based on the difficulty of finding discrete logs in finite fields, and more specifically on the Diffie and Hellman key exchange assumption.

• We will start by reviewing Diffie-Hallman, then move to Elgamal.
Diffie and Hellman Key Exchange (reminder)

- **Public parameters:** A large prime $p$ (1024 bits, say) and a primitive element $g$ in $\mathbb{Z}_p^*$.
- Alice chooses at random an integer $a$ from the interval $[0..p-2]$. She sends $x = g^a \pmod{p}$ to Bob (over the insecure channel).
- Bob chooses at random an integer $b$ from the interval $[0..p-2]$. He sends $y = g^b \pmod{p}$ to Alice (over the insecure channel).
- Alice, holding $a$, computes $y^a = (g^b)^a = g^{ba}$.
- Bob, holding $b$, computes $x^b = (g^a)^b = g^{ba}$.
- Now both have the shared secret, $g^{ba}$. 
Elgamal PKC (note resemblance to DH)

- Public information: A large prime \( p \), where \( p - 1 \) has a known factorization and a large prime factor. Recommended to take \( p = 2q + 1 \), where \( q \) is also a prime, and \( p \) is 756 or 1024 bits long.
  - A multiplicative generator \( g \) of \( \mathbb{Z}_p^* \)
  - Bob publishes \( p, g \).
  - Bob picks \( a \in [0..p - 2] \) at random.
  - Bob computes and publishes \( \beta = g^a \mod p \).
- Bob’s private information: \( a \).
- Encryption: of the message \( m \):
  - Alice picks \( k \in [0..p - 2] \) at random.
  - Alice computes \( g^k \mod p, m\beta^k \mod p \).
  - Alice sends \( E(m) = (g^k, m\beta^k) \) to Bob.
    (\( \beta^k \) “masks” \( m \); \( k \) obviously is not made public).
- Decryption of \( (g^k, m\beta^k) = (c_1, c_2) \):
  - Bob computes \( c_1^a = (g^k)^a = (g^a)^k = \beta^k \mod p \).
  - This enables Bob to compute the multiplicative inverse of \( \beta^k \mod p, \beta^{-k} \) (even though he does not know \( k \)).
  - Bob now computes \( \beta^{-k} \cdot c_2 = m \).
Properties of Elgamal Public Key Cryptosystem

- Encryption is randomized: $m \rightarrow (g^k, m^\beta^k)$.
- Alice should use a new, independent $k$ for every encryption.
- Even if same $m$ is sent twice, different $k$ must be used.

- Encryption takes two modular exponentiations.
- Decryption takes one modular exponentiation.
- Ciphertext, $(g^k, m^\beta^k)$, is twice as long as plaintext $m$. 
Properties of Elgamal Public Key Cryptosystem (2)

- Cryptosystem is **vulnerable** to chosen ciphertext attacks.
- Given $E(m) = (c_1, c_2) = (g^k, m \beta^k)$,
- Attacker chooses a random $s$, computes $(c_1, s \cdot c_2) = (g^k, s \cdot m \beta^k)$
- Attacker asks for decryption of $(c_1, s \cdot c_2)$, which equals $s \cdot m$, from which $m$ is easily recovered.

- Cryptosystem is **multiplicative**. Given $E(m) = (c_1, c_2) = (g^k, m \beta^k)$, $E(m') = (c_1', c_2') = (g^{k'}, m' \beta^{k'})$, can easily obtain $E(m \cdot m') = (c_1 c_1', c_2 c_2') = (g^{k+k'}, m \cdot m' \beta^{k+k'})$ (without knowing any secret information).
Does DH Key Exchange Hide All Partial Information? (reminder)

- From $g^a$ and $g^b$, Eve could easily deduce if $a$ and $b$ are even or odd. The exponent arithmetic is done modulo $p - 1$, which is even.
- If both $a$ and $b$ are odd, then $ab \pmod{p - 1}$ is odd too, and $g^{ba}$ is not a QR. If $a$, $b$, or both are even, then $ab \pmod{p - 1}$ is even, so $g^{ba}$ is a QR.
- Thus in (this original version) of DH key exchange, does leak some partial information – specifically the QR bit of the key $g^{ba}$.
- Same type of partial information is leaked in Elgammal encryption.
Does Elgamal Encryption Hides All Partial Information? (reminder)

- From $\beta = g^a$, Eve could easily deduce if $a$ is even or odd.
- From $g^k$, Eve could easily deduce if $k$ is even or odd.
- If both $a$ and $k$ are odd, then $ak \pmod{p-1}$ is odd too, and $\beta^k = g^{ak}$ is not a QR. If $a$, $b$, or both are even, then $ab \pmod{p-1}$ is even, so $g^{ak}$ is a QR.
- Thus from $m\beta^k = mg^{ak}$, Eve can deduce if $m$ is a QR or not a QR.

- So this type of partial information is leaked in Elgammal encryption as well.
Restricting the Message Space

- Standard fix for DH key exchange to this partial information leakage problem: \( p \) is chosen to be of the form \( p = 2q + 1 \), where \( q \) is a prime.
- Instead of working in \( \mathbb{Z}_p^* \), work with \( \text{QR} \), the quadratic residues of \( \mathbb{Z}_p^* \).
- \( \text{QR} \) is a cyclic group with exactly \( q \) elements.
- Instead of working with a multiplicative generator \( g \) of \( \mathbb{Z}_p^* \), work with a multiplicative generator \( h \) of \( \text{QR} \), the quadratic residues of \( \mathbb{Z}_p^* \).

- An identical fix is applicable to Elgamal PKC.
- Alice should now encode messages as quadratic residues.
- Encoding messages as QR elements is easiest if \(-1\) is not a QR in \( \mathbb{Z}_p^* \). We omit the details.
Signatures

http://lacourphoto.net/uploaded.images/signatures1-770492.jpg

Introduction to Modern Cryptography
Benny Chor
RSA: Review and Properties
Factoring Algorithms
Trapdoor One Way Functions
PKC Based on Discrete Logs (Elgamal)
Signature Schemes
Hand Written Signatures

- Relate an individual, through a handwritten signature, to a document.
- Signature can be verified against a prior authenticated one, which was signed in person in a bank, in the presence of a public notary public, etc.
- Should be hard to forge.
- Are legally binding (convince a third party, e.g. a judge).
Digital Signature Schemes

- Would like to achieve all features of hand written signatures, plus more.
- For example, should be able to base difficulty of forgery on some hard computational problem, not just on ineptitude of forger.
- Diffie and Hellman were first to propose such framework.
- To be continued.