

Introduction to Cryptography

Lecture 6

Public key encryption
RSA, Rabin

Benny Pinkas

Reminder: The Multiplicative Group Z_{pq}^*

- p and q denote two large primes (e.g. 512 bits long).
- Denote their product as $N = pq$.
- The multiplicative group $Z_N^* = Z_{pq}^*$ contains all integers in the range $[1, pq-1]$ that are relatively prime to both p and q .

- The size of the group is
 - $\varphi(n) = \varphi(pq) = (p-1)(q-1) = N - (p+q) + 1$
- For every $x \in Z_N^*$, $x^{(p-1)(q-1)} = 1 \pmod N$.

Reminder: RSA Public Key Cryptosystem

- Public key:
 - $N=pq$ the product of two primes
 - e such that $\gcd(e, \phi(N))=1$ (*are these hard to find?*)
- Private key:
 - d such that $de \equiv 1 \pmod{\phi(N)}$
- Encryption of $M \in \mathbb{Z}_N^*$
 - $C=E(M)=M^e \pmod{N}$
- Decryption of $C \in \mathbb{Z}_N^*$
 - $M=D(C)=C^d \pmod{N}$ (*why does it work?*)

Properties of RSA

- Deterministic encryption. In textbook RSA:
 - M is always encrypted as M^e
 - The ciphertext is as long as the domain of M
- The public exponent e may be small. It's common to choose its value to be either 3 or $\underbrace{2^{16}+1}$. The private key d must be long.
why?
 - Each encryption involves several modular multiplications. Decryption is longer.
- Chosen ciphertext attack: (homomorphic property)
 - Given a ciphertext $C=M^e$, choose a random R and generate $C'=CR^e$ (an encryption of $M\cdot R$). A decryption of C' reveals M .

Decryption overhead

- Usage of a small $e \Rightarrow$ Encryption is more efficient than a full blown exponentiation.
- Decryption requires a full exponentiation ($M=C^d \bmod N$)
- Can this be improved?

The Chinese Remainder Theorem (CRT)

- Thm:
 - Let $N=pq$ with $\gcd(p,q)=1$.
 - Then for every pair $(y,z) \in \mathbb{Z}_p \times \mathbb{Z}_q$ there exists a *unique* $x \in \mathbb{Z}_n$, s.t.
 - $x=y \pmod p$
 - $x=z \pmod q$
- Proof:
 - The extended Euclidian algorithm finds a,b s.t. $ap+bq=1$.
 - Define $c=bq$. $c=1 \pmod p$. $c=0 \pmod q$.
 - Define $d=ap$. $d=1 \pmod q$. $d=0 \pmod p$.
 - Let $x=cy+dz \pmod N$.
 - $cy+dz = 1y + 0 = y \pmod p$.
 - $cy+dz = 0 + 1z = z \pmod q$.
 - (How efficient is this?)
 - (Inverse: finding (y,z) from x is easy.)

More efficient RSA decryption

- CRT:
 - Given p, q compute a, b s.t. $ap + bq = 1$.
 - $c = bq$; $d = ap$
- Decryption, given C :
 - Compute $y' = C^d \bmod p$. (instead of d can use $d' = d \bmod p-1$)
 - Compute $z' = C^d \bmod q$. (instead of d can use $d'' = d \bmod q-1$)
 - Compute $M = cy' + dz' \bmod N$.
- Overhead:
 - Two exponentiations modulo p, q , instead of one exponentiation modulo N .
 - Overhead of exponentiation is cubic in length of modulus.
 - I.e., save a factor of $2^3/2$.

} Once for all messages

Security reductions

- Security by reduction
 - Define what it means for the system to be “secure” (chosen plaintext/ciphertext attacks, etc.)
 - State a “hardness assumption” (e.g., that it is hard to extract discrete logarithms in a certain group).
 - Show that if the hardness assumption holds then the cryptosystem is secure.
- Benefits:
 - To examine the security of the system it is sufficient to check whether the assumption holds
 - Similarly, for setting parameters (e.g. group size).

RSA Security

- If factoring N is easy then RSA is insecure
 - (factor $N \Rightarrow$ find $p, q \Rightarrow$ find $(p-1)(q-1) \Rightarrow$ find d from e)
- Factoring assumption:
 - For a randomly chosen p, q of appropriate length, it is infeasible to factor $N=pq$.
- This assumption might be too weak (might not ensure secure encryption)
 - Maybe it's possible to break RSA without factoring N ?
 - We don't know how to reduce RSA security to the hardness of factoring.
- Fact: finding d is equivalent to factoring.
 - I.e., if it is possible to find d given (N, e) , then it is easy to factor N .
- “hardness of finding d assumption” no stronger than hardness of factoring.

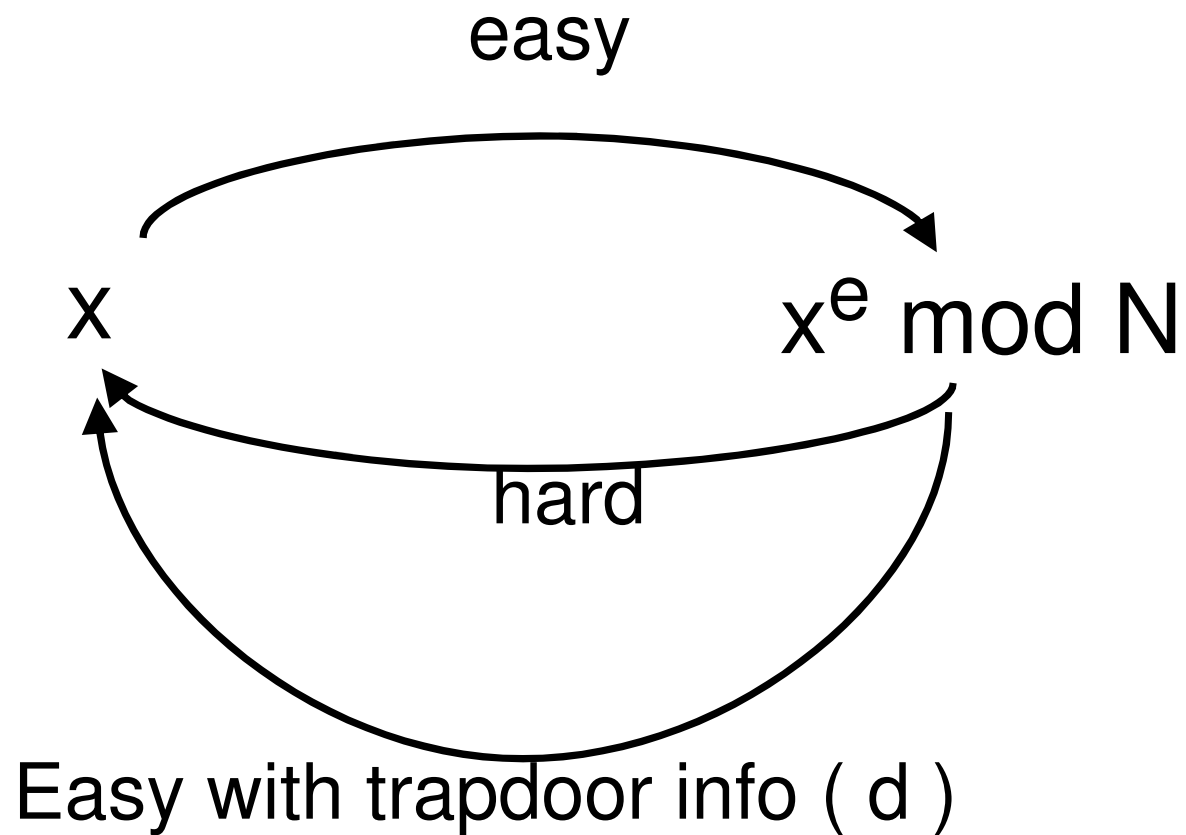
Side discussion: Is it safe to use common modulus ?

- Consider the following environment:
 - There is a global modulus N . No one knows its factoring.
 - Each party has a pair (e_i, d_i) , such that $e_i d_i = 1 \pmod{N}$.
 - Used as a public/private key pair.
- The system is insecure.
- Party 1, knowing (e_1, d_1)
 - can factor N
 - Find d_i for any other party i .

The RSA assumption: Trap-Door One-Way Function (OWF)

- (what is the minimal assumption required to show that RSA encryption is secure?)
- (Informal) definition: $f : D \rightarrow R$ is a *trapdoor one way function* if there is a trap-door s such that:
 - Without knowledge of s , the function f is a one way. I.e., for a randomly chosen x , it is hard to invert $f(x)$.
 - Given s , inverting f is easy
- Example: $f_{g,p}(x) = g^x \bmod p$ is *not* a trapdoor one way function.
- Example: assuming that RSA is a trapdoor OWF
 - $f_{N,e}(x) = x^e \bmod N$. (assumption: for a random N, e, x , inverting is hard.)
 - The trapdoor is d s.t. $ed = 1 \bmod \phi(N)$
 - $[f_{N,e}(x)]^d = x \bmod N$

RSA as a One Way Trapdoor Permutation



RSA assumption: cautions

- The RSA assumption is quite well established:
 - RSA is a Trapdoor One-Way Permutation
 - Hard to invert on random input – without secret key
- But is it a secure cryptosystem?
 - Given the assumption it is hard to reconstruct the input, but is it hard to learn *anything* about the input?
- Theorem [G]: RSA hides $\log(\log(n))$ least *and* most significant bits of a uniformly-distributed random input
 - But some (other) information about pre-image may leak
 - And... adversary can detect a repeating message

RSA with a small exponent

- Setting $e=3$ enables efficient encryption
- Might be insecure if not used properly
 - Assume three users with public keys N_1, N_2, N_3 .
 - Alice encrypts the same message to all of them
 - $C_1 = m^3 \bmod N_1$
 - $C_2 = m^3 \bmod N_2$
 - $C_3 = m^3 \bmod N_3$
- Can an adversary which sees C_1, C_2, C_3 find m ?
 - $m^3 < N_1 N_2 N_3$
 - N_1, N_2 and N_3 are most likely relatively prime (otherwise can factor).
 - Chinese remainder theorem \rightarrow can find $m^3 \bmod N$ (and therefore m^3 over the integers)
 - Easy to extract 3rd root over the integers.

Rabin's encryption systems

- Key generation:
 - Private key: random primes p, q (e.g. 512 bits long).
 - Public key: $N=pq$.
- Encryption:
 - Plaintext $m \in \mathbb{Z}_N^*$.
 - Ciphertext: $c = m^2 \bmod N$. (*very efficient*)
- Decryption: Compute $c^{1/2} \bmod N$.

Square roots modulo N

- \Rightarrow Let x be a quadratic residue (QR) modulo $N=pq$, then
 - $x \bmod p$ is a QR mod p . $x \bmod q$ is a QR mod q
 - $x \bmod p$ has *two* roots mod p : y and $p - y$
 - $x \bmod q$ has *two* roots mod q : z and $q - z$
- \Leftarrow If x is a QR mod p and mod q , it is a QR mod N . (Follows from the Chinese remainder theorem.)
 - Each combination of roots modulo p and q results in a root modulo N .
 - We get four roots modulo pq : $A, B, pq - A, pq - B$
 - $(y, z) \rightarrow A, \quad (p - y, q - z) \rightarrow pq - A$
 - $(y, q - z) \rightarrow B, \quad (p - y, z) \rightarrow pq - B$
 $\quad = (y, z) \cdot (1, -1)$

Square roots modulo N

- $N = pq$.
- If x has a square root modulo N then it has 4 different square roots modulo N .
 - Let A be s.t. $A^2 = x \pmod N$.
 - Let c be s.t. $c = 1 \pmod p$, $c = -1 \pmod q$.
 - Then A , $-A$, cA , $-cA$ are all square roots of x modulo N .
- Exactly $\frac{1}{4}$ of the elements are QR mod N .
- $QR_N = QR_p \times QR_q$. $|QR_N| = (p-1)(q-1)/4$
- Assume that $p = q = 3 \pmod 4$. (Blum integers.)
 - -1 is an NQR mod p and mod q .
 - Exactly one of the roots is a QR mod p and a QR mod q .
 - Similarly, for every combination of QR/NQR mod p and mod q .

Finding square roots modulo N

- Need to compute $y=x^{1/2} \bmod N$.
- Suppose we know p, q .
 - Compute the roots of x modulo p, q . Use Chinese remainder theorem to find x .
- Computing square roots in Z_p^* ,
 - Recall, $x \in QR_p$ iff $x^{(p-1)/2} = 1 \bmod p$.
 - Assume $p \equiv 3 \pmod{4}$. (p is a Blum integer).
 - Compute the root as $y = x^{(p+1)/4} \bmod p$.
 - $(p+1)/4$ is an integer
 - $y^2 = (x^{(p+1)/4})^2 = x^{(p+1)/2} = x^{(p-1)/2}x = x$
 - If $p \equiv 1 \pmod{4}$ the computation is more complicated (no deterministic algorithm is known)

Decryption of Rabin cryptosystem

- Input: c, p, q . ($p=q=3 \pmod{4}$)
- Decryption:
 - Compute $m_p = c^{(p+1)/4} \pmod{p}$.
 - Compute $m_q = c^{(q+1)/4} \pmod{q}$.
 - Use CRT to compute the four roots mod N , i.e. four values mod N corresponding to $[m_p, p-m_p] \times [m_q, q-m_q]$
- There are four possible options for the plaintext!
 - The receiver must select the correct plaintext
 - This can be solved by requiring the sender to embed some redundancy in m
 - E.g., a string of bits of specific form

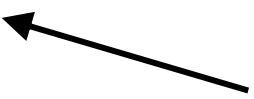
Security of the Rabin cryptosystem

- The Rabin cryptosystem is secure against passive attacks iff factoring is hard. 😊
- The Rabin cryptosystem is completely insecure against chosen-ciphertext attacks 😞

Security of the Rabin cryptosystem

- Security against chosen plaintext attacks
- Suppose there is an adversary that breaks the system
 - Adversary's input: N, c
 - Adversary's output: m s.t. $m^2 = c \pmod N$.
- We show a reduction showing that given this adversary we can break the factoring assumption.
- I.e., we build an algorithm:
 - Input: N
 - Operation: can ask queries to the Rabin decryption oracle
 - Output: the factoring of N .
- Therefore, if one can break Rabin's cryptosystem it can also solve factoring.
- Therefore, if factoring is hard the Rabin cryptosystem is "secure".

The reduction

- Input: N
- Operation:
 - Choose random x .
 - Send N and $c=x^2 \bmod N$, to adversary.
 - Adversary answers with y s.t. $c=y^2 \bmod N$.
 - If $y=x$ or $y=N-x$, go back to step 1. 
 - Otherwise
 - $x^2 - y^2 = 0 \bmod N$.
 - $0 \neq (x-y)(x+y) = cN = cpq$.
 - Compute $\gcd(x+y, N)$ and obtain p or q .
 - (suppose $x \neq y \bmod p$. Then $x = -y \bmod p$, and $x+y = cp$.)

happens with
prob 1/2

Insecurity against chosen-ciphertext attacks

- A chosen-ciphertext attack reveals the factorization of N .
- The attacker's challenge is to decrypt a ciphertext c .
- It can ask the receiver to decrypt any ciphertext except c .
- The attacker can use the receiver as the “adversary” in the reduction, namely
 - Chooses a random x and send $c=x^2 \bmod N$ to the receiver
 - The receiver returns a square root y of c
 - With probability $1/2$, $x \neq y$ and $x \neq -y$. In this case the attacker can factor N by computing $\gcd(x-y, N)$.