Introduction to Cryptography
Lecture 7

Public key cryptography

Benny Pinkas

Last lecture

- Basic number theory
  - Lots of facts about groups
- In particular
  - Z_p^* Multiplication modulo a prime number p
    - $(G, \circ) = (\{1,2,\ldots,p\text{-}1\}, \times)$, e.g., $Z_7^* = (\{1,2,3,4,5,6\}, \times)$.
  - Z_N^* Multiplication modulo a composite number N
    - $(G, \circ) = (\{a \text{ s.t. } 1 \le a \le N\text{-}1$ and $gcd(a,N)=1\}, \times)$
    - *E.g.*, $Z_{10}^* = (\{1,3,7,9\}, \times)$
  - A group G is cyclic if there exists a generator g, s.t. $\forall a \in G$, $\exists\, i$ s.t. $g^i=a$.

The Diffie-Hellman Key Exchange Protocol

- Public parameters: a group where the DDH assumption holds. For example, Z_p* (where $|p|$= *768* or *1024, p=2q+1*), and a generator g of $H \subset Z_p$* of order q.

- Alice:
  - picks a random $a \in [1,q]$.
  - Sends g^a mod p *to Bob.*
  - *Computes* $k=(g^b)^a$ mod p

- Bob:
  - picks a random $b \in [1,q]$.
  - Sends g^b mod p *to Bob.*
  - *Computes* $k=(g^a)^b$ mod p

- $K = g^{ab}$ is used as a shared key between Alice and Bob.
  - DDH assumption $\Rightarrow$ K is indistinguishable from a random key

Diffie-Hellman: security

- A *(passive)* adversary
  - Knows Z_p^*, g
  - Sees g^a, g^b
  - Wants to compute g^{ab}, or at least learn something about it
- Recall the Decisional Diffie-Hellman problem:
  - Given random $x,y \in Z_p^*$, such that $x=g^a$ and $y=g^b$; *and a pair* (g^{ab},g^c) (in random order, for a random c), it is hard to tell which is g^{ab}.
  - An adversary that distinguishes the key g^{ab} generated in a DH key exchange from random, can also break the DDH.

  - *Note:* it is insufficient to require that the adversary cannot compute g^{ab}.

Diffie-Hellman key exchange: usage

- The DH key exchange can be used in any group in which the Decisional Diffie-Hellman (DDH) assumption is believed to hold.
- Currently, Z_p^* and elliptic curve groups.

- Common usage:
  – Overhead: 1-2 exponentiations
  – Usually,
    - A DH key exchange for generating a master key
    - Master key used to encrypt session keys
    - Session key is used to encrypt traffic with a symmetric cryptosystem

An active attack against the Diffie-Hellman Key Exchange Protocol

- An active adversary Eve.
- Can read and change the communication between Alice and Bob.
- …As if Alice and Bob communicate via Eve.

Man–in-the-Middle: an active attack against the Diffie-Hellman Key Exchange protocol

- Alice:
  - picks a random $a \in [1,q]$.
  - Sends g^a mod p *to Bob.*

- Bob:

Eve changes g^a to g^c

  - picks a random $b \in [1,q]$.
  - Sends g^b mod p to Alice.

Eve changes g^b to g^d

  - *Computes* $k=(g^d)^a$ mod p
  - *Computes* $k=(g^c)^b$ mod p

Keys:

Alice	Eve	Bob
g^{ad}	g^{ad}, g^{bc}	g^{bc}

  - Solution: ? (wireless usb)

Public key encryption

- Alice publishes a *public* key PK_{Alice}.
- Alice has a *secret* key SK_{Alice}.
- Anyone knowing PK_{Alice} can encrypt messages using it.
- Message decryption is possible only if SK_{Alice} is known.

- Compared to symmetric encryption:
  - Easier key management: n users need n keys, rather than $O(n^2)$ keys, to communicate securely.
- Compared to Diffie-Hellman key agreement:
  - No need for an interactive key agreement protocol. (Think about sending email…)

- Secure as long as we can trust the association of keys with users.

Public key encryption

- Must have different keys for encryption and decryption.
- Public key encryption cannot provide perfect secrecy:
  - Suppose $E_{pk}()$ is an algorithm that encrypts m=0/1, and uses r random bits in operation.
  - An adversary is given $E_{pk}(m)$. It can compare it to all possible 2^r encryptions of 0…
- Efficiency is the main drawback of public key encryption.

Defining a public key encryption

- The definition must include the following algorithms;
- Key generation: KeyGen(1^k)$\rightarrow$(PK,SK) (where k is a security parameter, e.g. k=1000).
- Encryption: C = E_{PK}(m) (E might be a randomized algorithm)
- Decryption: M= D_{SK}(C)

The El Gamal public key encryption system

- Public information (can be common to different public keys):
  - A group in which the DDH assumption holds. Usually start with a prime $p=2q+1$, and use $H \subset Z_p^*$ of order q. Define a generator g of H.

- Key generation: pick a random private key a in [1,|H|] (e.g. $0<a<q$). Define the public key $h=g^a$ ($h=g^a \bmod p$).

- Encryption of a message $m \in H \subset Z_p^*$
  - Pick a random $0 < r < q$.
  - The ciphertext is $(g^r, h^r \cdot m)$.

  } Using public key alone

- Decryption of (s,t)
  - Compute t / s^a ($m = h^r \cdot m / (g^r)^a$)

  } Using private key

El Gamal and Diffie-Hellman

- ElGamal encryption is similar to DH key exchange
  - DH key exchange: Adversary sees g^a, g^b. Cannot distinguish the key g^{ab} *from random.*
  - El Gamal:
    - A fixed public key g^a.
    - Sender picks a random g^r.

      Known to the adversary
    - Sender encrypts message using g^{ar}.

      Used as a key

- El Gamal is like DH where
  - The same g^a is used for all communication
  - There is no need to explicitly send this g^a (it is already known as the public key of Alice)

Semantic security

- Semantic Security: knowing that an encryption is either $E(m_0)$ or $E(m_1)$, (where m_0,m_1 are known) an adversary cannot decide with probability better than ½ which is the case.

- Suppose that a public key encryption system is deterministic., then it cannot have semantic security.
  - Namely, E(m) is a deterministic function of m and P.
  - Then if Eve suspects that Bob might encrypt either m_0 or m_1, she can compute (by herself) $E(m_0)$ and $E(m_1)$ and compare them to the encryption that Bob sends.

El Gamal encryption: breaking semantic security implies breaking DDH

- Proof by reduction:
  - We are given $(g, g^a, g^b, (D_1, D_2))$ where one of D_1, D_2 is g^{ab}, and the other is g^r. We need to identify g^{ab}.
  - We give the adversary g and a public key: $h=g^a$.
  - The adversary chooses m_0, m_1.
  - We give the adversary $(g^b, D_e \cdot m_c)$, where c, e are random.
  - If the adversary guesses c correctly, we decide that $D_e = g^{ab}$. Otherwise we decide that $D_e = g^r$.
- Analysis:
  - Suppose that the adversary can guess c with prob ¾.
  - If $D_e = g^{ab}$ then the adversary finds c with probability ¾, otherwise it finds c with probability ½.
  - Our success probability $\frac{1}{2} \cdot \frac{3}{4} + \frac{1}{2} \cdot \frac{1}{2} = 5/8$.

The El Gamal public key encryption system

- Setting the public information
- *A large prime p, and a generator g of* $H \subset Z_p^*$ of order q.
  - $|p|$ = 756 or 1024 bits.
  - $p-1$ must have a large prime factor (e.g. $p=2q+1$)
    - Otherwise it is easy to solve discrete logs in Z_p^* (relevant also to DH key agreement)
    - Needed for the DDH assumption to hold (Legendre's symbol)
  - g must be a generator of a large subgroup of Z_p^*.

- Encoding the message:
  - m must be in the subgroup generated by g.
  - Alternatively, encrypt m using $(g^r, H(h^r) \oplus m)$. *Decryption is done by computing* $H((g^r)^a)$. (H is a hash function that preserves the pseudo-randomness of h^r.)

The El Gamal public key encryption system

- Overhead:
  - Encryption: two exponentiations; preprocessing possible.
  - Decryption: one exponentiation.
  - message expansion: $m \Rightarrow (g^r, h^r \cdot m)$.

- Randomized encryption
  - Must use fresh randomness *r* for every message.
  - Two different encryptions of the same message are different! (provides semantic security)

Homomorphic property

- Insecurity against chosen ciphertext attacks:
  - Attacker wants to decrypt $(s,t) = (g^r, h^r \cdot m)$.
  - Chooses random r', computes $(s',t')=(s, t\cdot r') = (g^r, h^r\cdot(m\cdot r'))$.
  - Asks for a decryption of (s',t'). Receives $m\cdot r'$.

- Homomorphic property:
  - Given encryptions of x,y, it's easy to generate an encryption of $x\cdot y$.
    - $(g^r, h^r\cdot x) \times (g^{r'}, h^{r'}\cdot y) \rightarrow (g^{r''}, h^{r''} \cdot x\cdot y)$

Homomorphic encryption

- Homomorphic encryption is useful for performing operations over encrypted data.
- Given $E(m_1)$ and $E(m_2)$ it is easy to compute $E(m_1 m_2)$.

- For example, an election procedure:
  – A “Yes” is E(2). A “No” vote is E(1).
  – Take all the votes and multiply them. Obtain $E(2^j)$, where j is the number of “Yes” votes.
  – Decrypt the result and find out how many “Yes” votes there are, without identifying how each person voted.

Integer Multiplication & Factoring as a One Way Function.

$$\begin{tikzcd} p,q \arrow[r, bend left, "\textbf{easy}"] & N=pq \arrow[l, bend left, "\textbf{hard}"] \end{tikzcd}$$

Can a public key system be based on this observation ?????

Excerpts from RSA paper (CACM, 1978)

The era of "electronic mail" may soon be upon us; we must ensure that two important properties of the current "paper mail" system are preserved: (a) messages are *private*, and (b) messages can be *signed*. We demonstrate in this paper how to build these capabilities into an electronic mail system.

At the heart of our proposal is a new encryption method. This method provides an implementation of a "public-key cryptosystem," an elegant concept invented by Diffie and Hellman. Their article motivated our research, since they presented the concept but not any practical implementation of such system.

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
  - $\phi(n) = \phi(pq) = (p-1)(q-1) = N - (p+q) + 1$

- For every $x \in Z_N^*$, $x^{\phi(N)} = x^{(p-1)(q-1)} = 1 \bmod N$.

Exponentiation in Z_N^*

- Motivation: use exponentiation for encryption.

- Let e be an integer, $1 < e < \phi(N) = (p-1)(q-1)$.
  - Question: When is exponentiation to the e^{th} power, ($x \to x^e$), a one-to-one operation in Z_N^* ?

- Claim: If e is relatively prime to $(p-1)(q-1)$ then $x \to x^e$ is a one-to-one operation in Z_N^*.
- Constructive proof:
  - Since gcd$(e, (p-1)(q-1))=1$, e has a multiplicative inverse modulo $(p-1)(q-1)$.
  - Denote it by d, then $ed=1+c(p-1)(q-1)=1+c\phi(N)$.
  - Let $y=x^e$, then $y^d = (x^e)^d = x^{1+c\phi(N)} = x$.
  - I.e., $y \to y^d$ is the inverse of $x \to x^e$.

The RSA Public Key Cryptosystem

- Public key:
  - $N=pq$ the product of two primes (we assume that factoring N is hard)
  - e such that $\gcd(e,\phi(N))=1$ *(are these hard to find?)*
- Private key:
  - d such that $de\equiv 1 \bmod \phi(N)$

- Encryption of $M\in Z_N{}^*$
  - $C=E(M)=M^e \bmod N$
- Decryption of $C\in Z_N{}^*$
  - $M=D(C)=C^d \bmod N$ *(why does it work?)*

Constructing an instance of the RSA PKC

- Alice
  - picks at random two large primes, *p* and *q*.
  - picks (uniformly at random) a (large) *d* that is relatively prime to (p-1)(q-1) (namely, gcd$(d,\phi(N))=1$).
  - Alice computes *e* such that $de \equiv 1 \bmod \phi(N)$

- Let $N=pq$ be the product of *p* and *q*.
- Alice publishes the public key (N,e).
- Alice keeps the private key *d*, as well as the primes *p, q* and the number $\phi(N)$, in a safe place.

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

- Corolalry: RSA is does not have semantic security.

- Chosen ciphertext attack: (homomorphic property)
  - RSA is susceptible to chosen ciphertext attacks:
  - Given a ciphertext $C=M^e$, choose a random R and generate $C'=CR^e$ (an encryption of $M \cdot R$). Decrypting C' reveals M.

Efficiency

- The public exponent *e* may be small.
  - It is common to choose its value to be either 3 or $2^{16}+1$. The private key *d* must be long.
  - Each encryption involves only a few modular multiplications. Decryption requires a full exponentiation.

- 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 $(y,z) \in Z_p \times Z_q$ there exists a *unique* $x \in Z_n$, s.t.
    - $x=y \bmod p$
    - $x=z \bmod q$
- Proof:
  - The extended Euclidian algorithm finds a,b *s.t.* $ap+bq=1$.
  - Define $c=bq$. $c=1 \bmod p$. $c=0 \bmod q$.
  - Define $d=ap$. $d=0 \bmod p$. $d=1 \bmod q$.
  - *Let* $x=cy+dz \bmod N$.
    - $cy+dz = 1y + 0 = y \bmod p$.
    - $cy+dz = 0 + 1z = z \bmod q$.
  - (How efficient is this?)
  - (The inverse operation, finding (y,z) from x, is easy.)

More efficient RSA decryption

- CRT:
  - Given p,q compute a,b s.t. $ap+bq=1.$
  - $c=bq;\ \ d=ap$

Once for all messages

- Decryption, given C:
  - Compute $y'=C^d \bmod p$. (instead of d can use $d'=d \bmod p\text{-}1$)
  - Compute $z'=C^d \bmod q$. (instead of d can use $d''=d \bmod q\text{-}1$)
  - 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 $2^3/2$.

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).

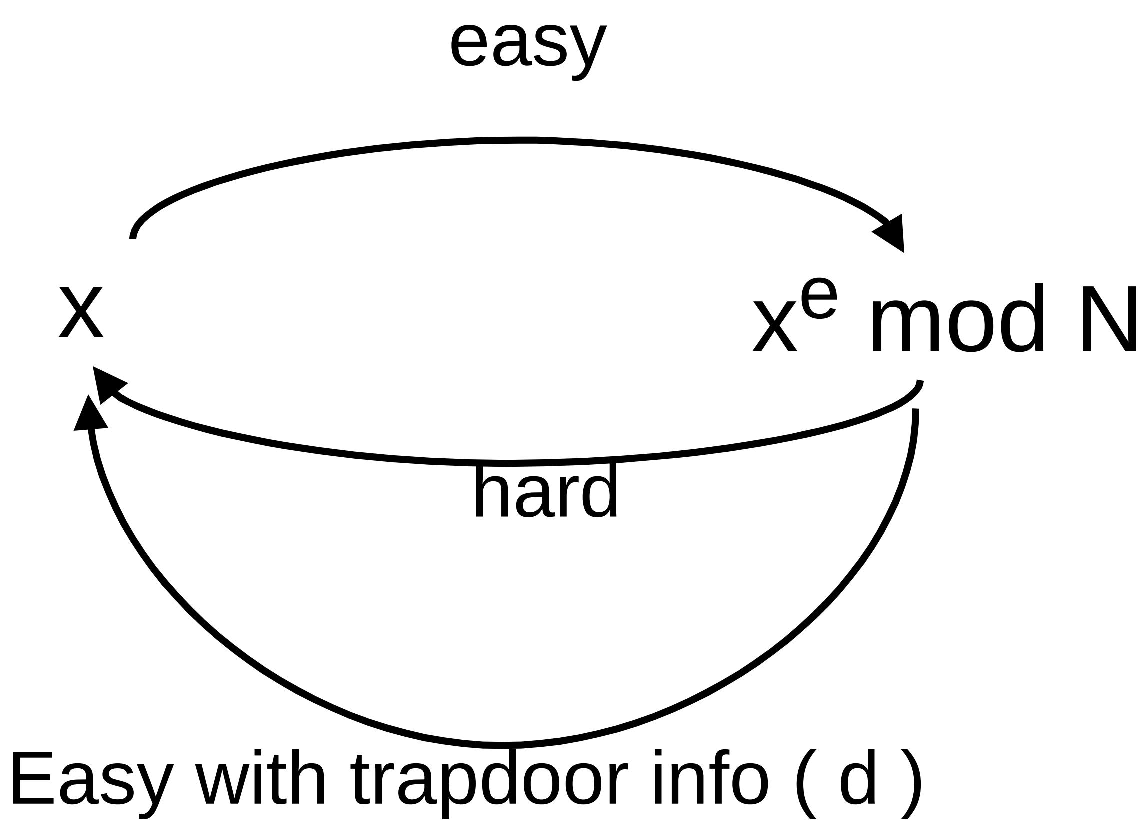
RSA Security

- 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.

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. $ed = 1 \bmod \varphi(N)$
  - $[f_{N,e}(x)]^d = x \bmod N$

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 the 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

Is it safe to use a 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 $e_i,d_i = 1$ mod 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.

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 -> can find $m^3 \bmod N$ (and therefore m^3 over the integers)
  - Easy to extract 3rd root over the integers.