Introduction to Cryptography

Lecture 9

Benny Pinkas

Hard problems in cyclic groups of prime order

- The following problems are believed to be hard in subgroups of prime order of Z_p^* (if the subgroup is large enough)
  - The discrete log problem
  - The Diffie-Hellman problem: The input contains g and $x,y \in G$, such that $x=g^a$ and $y=g^b$ (where a,b were chosen at random). The task is to find $z=g^{a \cdot b}$.
  - The Decisional Diffie-Hellman problem: The input contains $x,y \in G$, such that $x=g^a$ and $y=g^b$ (and a,b were chosen at random); and a pair (z,z') where one of (z,z') is $g^{a \cdot b}$ and the other is g^c (for a random c). The task is to tell which of (z,z') is $g^{a \cdot b}$.
- Solving DDH $\leq$ solving DH $\leq$ solving DL
  - All believed to be hard if the size of the subgroup > 2^{700}.

The Diffie-Hellman Key Exchange Protocol

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

- 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}.
  - *Note:* We showed last week that the Diffie-Hellman key exchange is insecure against an active adversary.

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.

Notes on public key encryption

- Must use 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)

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)
    - This large prime factor is also needed for the DDH assumption to hold (Legendre's symbol).
  - g must be a generator of a large subgroup of Z_p^*, in which the DDH assumption holds.

The El Gamal public key encryption system

- Encoding the message:
  - m must be in the subgroup H generated by g.
  - If $p=2q+1$, and H is the subgroup of quadratic residues (which has $(p-1)/2=q$ items), we can map each message $m \in \{1,\ldots,(p-1)/2\}$ to the value $m^2 \bmod p$, which is in H.
    - Encrypt m^2 instead of m. Therefore decryption yields m^2 and not m. Must then compute a square root to obtain m.
  - 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: $m \Rightarrow (g^r, h^r \cdot m)$.

- This is a randomized encryption
  - Must use fresh randomness *r* for every message.
  - Two different encryptions of the same message are different! (this is crucial in order to provide semantic security)

Security proof

- 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.
  - Usually prove security by showing that breaking the cryptosystem means that the hardness assumption is false.
- 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).

Semantic security

- Semantic Security: knowing that an encryption is either $E(m_0)$ or $E(m_1)$, (where m_0,m_1 are known, or even chosen by the attacker) an adversary cannot decide with probability better than ½ which is the case.
  - This is a very strong security property.

- Suppose that a public key encryption system is deterministic., then it cannot have semantic security.
  - In this case, E(m) is a deterministic function of m and P.
  - Therefore, 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.

Goal and method

- Goal
  - Show that if the DDH assumption holds
  - then the El Gamal cryptosystem is semantically secure
- Method:
  - Show that if the El Gamal cryptosystem is *not* semantically secure
  - Then the DDH assumption *does not* hold

El Gamal encryption: breaking semantic security implies breaking DDH

- Proof by reduction:
  - We can use an adversay that breaks El Gamal.
  - We are given a DDH challenge: $(g, g^a, g^r, (D_0, D_1))$ where one of D_0, D_1 is g^{ar}, and the other is g^c. We need to identify g^{ar}.
  - We give the adversay g and a public key: $h=g^a$.
  - The adversary chooses m_0, m_1.
  - We give the adversay $(g^r, D_e \cdot m_b)$, using random $b, e \in \{0,1\}$.
    (That is, choose m_b randomly from $\{m_0, m_1\}$, choose D_e randomly from $\{D_0, D_1\}$. The result is a valid El Gamal encryption if $D_e = g^{ar}$.)
  - If the adversay guesses b correctly, we decide that $D_e = g^{ar}$. Otherwise we decide that $D_e = g^c$.

El Gamal encryption: breaking semantic security implies breaking DDH

- Analysis:
  - Suppose that the adversary can break the El Gamal encryption with prob 1.
  - If $D_e = g^{ar}$ then the adversary finds c with probability 1, otherwise it finds c with probability ½.
  - Our success probability $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{2} = 3/4$.

  - Suppose now that the adversary can break the El Gamal encryption with prob ½+p.
  - If $D_e = g^{ar}$ then the adversary finds c with probability ½+p, otherwise it finds c with probability ½.
  - Our success probability $\frac{1}{2} \cdot (\frac{1}{2}+p) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2}+\frac{1}{2}p$. QED

Chosen ciphertext attacks

- In a chosen ciphertext attack, the adversary is allowed to obtain decryptions of arbitrary ciphertexts of its choice (except for the specific message it needs to decrypt).
- El Gamal encryption is insecure against chosen ciphertext attacks:
  - Suppose the adversary wants to decrypt $<c_1,c_2>$ which is an ElGamal encryption of the form $(g^r,h^r m)$.
  - The adversary computes $c'_1=c_1 g^{r'}$, $c'_2=c_2 h^{r'} m'$, where it chooses r',m' at random.
  - It asks for the decryption of $<c'_1,c'_2>$. It multiplies the plaintext by $(m')^{-1}$ and obtains m.

Homomorphic property

- The attack on chosen ciphertext security is based on the homomorphic property of the encryption

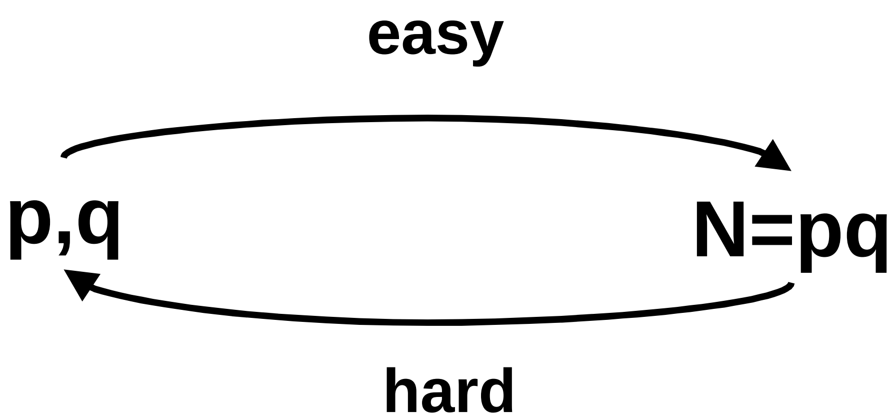
- Homomorphic property:
  - Given encryptions of x,y, it is easy to generate an encryption of $x \cdot y$
    - $(g^r, h^r \cdot x) \times (g^{r'}, h^{r'} \cdot y) \rightarrow (g^{r''}, h^{r''} \cdot x \cdot y)$

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 $E(m_1m_2)$, even if you don't know how to decrypt.

- 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 only 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.



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, ($x \rightarrow x^e$), a one-to-one operation in Z_N^* ?

- Claim: If e is relatively prime to $(p-1)(q-1)$ (namely $gcd(e, (p-1)(q-1))=1)$ then $x \rightarrow x^e$ 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., $y \rightarrow y^d$ is the inverse of $x \rightarrow x^e$.

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?)*