Introduction to Cryptography

Lecture 8

Benny Pinkas

Public Key-Exchange

- Goal: Two parties who do not share any secret information, perform a protocol and derive the same shared key.

- No eavesdropper can obtain the new shared key (if it has limited computational resources).

- The parties can therefore safely use the key as an encryption key.

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.

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

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 (against chosen plaintext attacks)

- 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.
- More precisely:
  - We generate a public key *PK* and give it to the adversary.
  - The adversary outputs two messages m_0, m_1.
  - We choose a random bit *b*, and give the ciphertext $E(m_b)$ to the adversary.
  - Adversary outputs a "guess" *b'*. It succeeds if *b'=b*.
  - The encryption scheme is semantically secure if $|Prob(b'=b) - ½|$ is negligible (as a function of the key length) for any polynomial adversary.

Semantic security

- This is a very strong security property. The adversary cannot even distinguish the encryption of two messages of its choice.
- Aka “security in the sense of indistinguishability”.

- Note that given the public key the adversary can generate encryptions of any message that it chooses.
- Deterministic public key encryption?
- 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 b with probability 1, otherwise it finds b 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 b with probability ½+p, otherwise it finds b 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

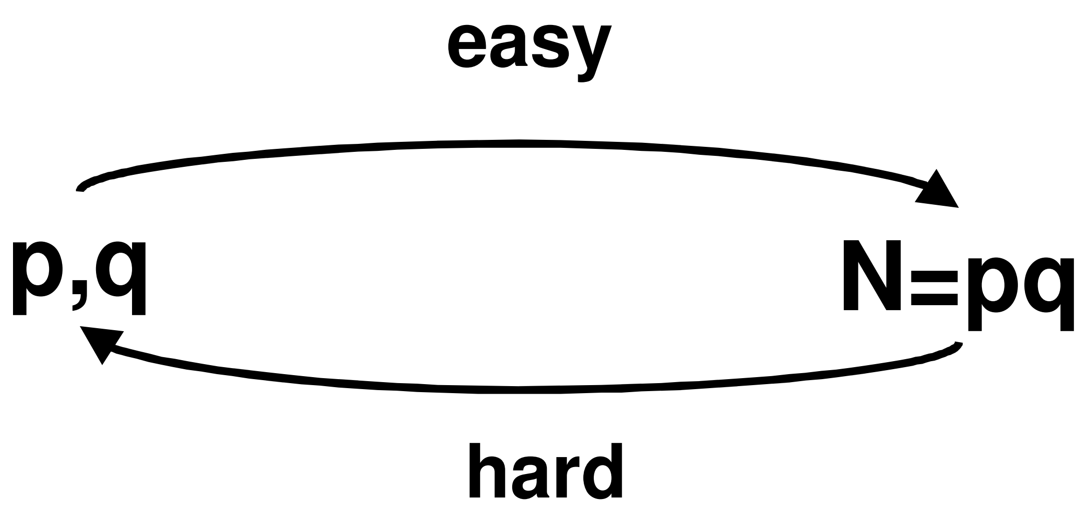
- Homomorphic property:
  - Given encryptions of x,y, it is easy to generate an encryption of $x \cdot y$
    - $(g^r, h^r \cdot x) \times (g^{r'}, h^{r'} \cdot y) \rightarrow (g^{r''}, h^{r''} \cdot x \cdot y)$

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 $de\equiv 1$ mod $\phi(N)$
- Encryption of $M\in Z_N^*$
  - $C=E(M)=M^e$ mod N
- Decryption of $C\in Z_N^*$
  - $M=D(C)=C^d$ mod 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 $de\equiv 1$ mod $\phi(N)$

- 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.

A small example

- Let $p=47$, $q=59$, $N=pq=2773$. $\phi(N)=46\cdot 58=2668$.

- Pick $e=17$. Since $157\cdot 17-2668=1$, then $d=157$.
- $e=17$ is 10001 in binary.

- To encrypt a message m, compute

  $$m^{17} = (((m^2)^2)^2)^2 \cdot m \mod 2773$$

  Decryption is less efficient

Efficiency

- The public exponent *e* may be small.
  - Instead of choosing a random *d* and setting e to be its inverse, it is common to choose the public exponent *e* to be either 3 or $2^{16}+1$. The private key *d* must be long.
  - Now, 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 $c=bq$. Therefore $c=1 \bmod p$. $c=0 \bmod q$.
  - Define $d=ap$. Therefore $d=0 \bmod p$. $d=1 \bmod q$.
  - *Let* $x=cy+dz \bmod N$.
    - $cy+dz = 1y + 0 = y \bmod p$.
    - $cy+dz = 0 + 1z = z \bmod q$.
  - (The inverse operation, finding (y,z) from x, is easy.)
  - (How efficient is this? Why is there a unique such $x \in Z_n$?)

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 $2^3/2$.

RSA with a small exponent

- Setting $e=3$ enables efficient encryption
- Might be insecure if not used properly
  - Assume that the message is short, for example $|M|<|N|/3$
  - In this case, $M^3 < N$, and therefore $M^3 \bmod N = M^3$ (over the integers).
  - For example, suppose that $M=10$. In this case $M^3 \bmod N = 1000$. (If $N>1000$.)
  - Extracting roots over the integers is easy, and therefore it is easy to find M.

RSA with a small exponent

- Another security problem with using short exponents (for example, $e=3$)
- Assume three users with public keys N_1, N_2, N_3.
  - Alice encrypts the same (long) 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?
  - $m^3 < N_1N_2N_3$
  - N_1, N_2 and N_3 are most likely relatively prime (otherwise can factor).
  - Chinese remainder theorem -> can find $m^3 \bmod N_1N_2N_3$ (and therefore m^3 over the integers)
  - Easy to extract 3rd root over the integers.

Random self reducibility of RSA

- Let *(N,e)* be an RSA public key.
- Suppose that there is a deterministic polynomial algorithm *A* running in time $|N|^C$ which on input $E(x)=x^e$ mod *N* outputs *x* for a fraction of ε of the inputs.

- Then *A* can be converted to a randomized algorithm *R*, which runs in expected time $|N|^C / \varepsilon$, which on input $E(x)=x^e$ mod *N* outputs *x* for <u>all</u> inputs.

- Proof (on board): easy.
- Corollary: For any (N,e), inverting RSA is either hard for all inputs or easy for all inputs.