Introduction to Cryptography

Lecture 8

Benny Pinkas

Groups we will use

- Z_p^* Multiplication modulo a prime number p
  - $(G, \circ) = (\{1,2,\ldots,p-1\}, \times)$
  - E.g., $Z_7^* = (\{1,2,3,4,5,6\} , \times)$

- Z_N^* Multiplication modulo a composite number N
  - $(G, \circ) = (\{a$ s.t. $1 \le a \le N-1$ and $gcd(a,N)=1\}, \times)$
  - E.g., $Z_{10}^* = (\{1,3,7,9\}, \times)$

Cyclic Groups

- Exponentiation is repeated application of $\circ$
  - $a^3 = a \circ a \circ a.$
  - $a^0 = 1.$
  - $a^{-x} = (a^{-1})^x$
- A group G is cyclic if there exists a generator g, s.t. $\forall a \in G, \exists i$ s.t. $g^i = a$.
  - I.e., G= <g> = $\{1, g, g^2, g^3, \ldots\}$
  - For example $Z_7^* = <3> = \{1,3,2,6,4,5\}$
- Not all $a \in G$ are generators of G, but they all generate a subgroup of G.
  - E.g. 2 is not a generator of Z_7^*
- The order of a group element a is the smallest $j>0$ s.t. $a^j=1$
- *Lagrange's theorem* $\Rightarrow$ for $x \in Z_p^*$, $ord(x) \mid p-1.$

Computing in Z_p^*

- P is a huge prime (1024 bits)
- Easy tasks (measured in bit operations):
  - Adding in O(log p) (namely, linear n the length of p)
  - Multiplying in O($\log^2$ p) (and even in O($\log^{1.7}$ p))
  - Inverting (a to a^{-1}) in O($\log^2$ p)
  - Exponentiations:
    - x^r mod p in O(log r · $\log^2$ p), using repeated squaring

Euler's phi function

- Lagrange's Theorem: $\forall a$ in a finite group G, $a^{|G|}=1.$
- Euler's phi function (aka, Euler's totient function),
  - $\phi(n)$ = number of elements in Z^*_n (i.e. | $\{x \mid gcd(x,n)=1, 1\le x\le n\}$ |
  - $\phi(p) = p-1$ for a prime $p.$
  - $n=\prod_{i=1..k} p_i^{e(i)} \Rightarrow \phi(n) = n\cdot\prod_{i=1..k} (1-1/p_i)$
  - $\phi(p^2) = p(p-1)$ for a prime $p.$
  - $n=p\cdot q \Rightarrow \phi(n) =(p-1)(q-1)$

- Corollary: For Z_n^* ($n=p\cdot q$), $|Z_n^*|= \phi(n) =(p-1)(q-1).$
- $\forall a\in Z_n^*$ it holds that $a^{\phi(n)}=1$ mod n
  - For Z_p^* (prime p), $a^{p-1}=1$ mod p (Fermat's theorem).
  - For Z_n^* ($n=p\cdot q$), $a^{(p-1)(q-1)}=1$ mod n

Finding prime numbers

Finding prime numbers

- Prime number theorem: #{primes $\leq x$} $\approx x / \ln x$ as $x \to \infty$
- How can we find a random k-bit prime?
  - Choose x at random in $\{2^k,\ldots,2^{k+1}-1\}$
    - (How many numbers in that range are prime?

      About $2^{k+1}/ \ln 2^{k+1} - 2^k/ \ln 2^k$ numbers, i.e. a $1/ \ln(2^k)$ fraction.)
  - Test if x is prime
    - (more on this later in the course)
- The probability of success is $\approx 1/\ln(2^k) = O(1/k)$.
- The expected number of trials is $O(k)$.

Finding generators

- How can we find a generator of Z_p^*?
- Pick a random number $a \in [1,p-1]$, check if is a generator
  - Naively, check whether $\forall\, 1 \le i \le p-2 \quad a^i \neq 1$ ☹
  - But we know that if $a^i=1 \bmod p$ then $i \mid p-1$.
  - Therefore need to check only i for which $i \mid p-1$.

- Easy if we know the factorization of $(p-1)$. In that case
  - For all $a \in Z_p^*$, the order of a divides $(p-1)$
  - For every integer divisor b of $(p-1)$, check if $a^b=1$ mod p.
  - If none of these checks succeeds, then a is a generator, since a is a generator iff $ord(a)=p-1$.

Finding prime numbers of the right form

- How can we know the factorization of p-1?
- Easy, for example, if *p=2q+1*, and *q* is prime.
- How can we find a *k*-bit prime of this form?

1. Search for a prime number *q* of length *k-1* bits. (Will be successful after about *O(k)* attempts.)
2. Check if *2q+1* is prime (we will see how to do this later in the course).
3. If not, go to step 1.

Hard problems in cyclic groups

A hard problem can be useful for constructing cryptographic systems, if we can show that breaking the system is equivalent to solving this problem.

The Discrete Logarithm

- Let G be a cyclic group of order q, with a generator g.
  - $\forall h \in G$, $\exists x \in [1,\ldots,q]$, such that $g^x = h$.
  - This x is called the discrete logarithm of h to the base g.

  - $\log_g h = x$.
  - $\log_g 1 = 0$, and $\log_g(h_1 \cdot h_2) = \log_g(h_1) + \log_g(h_2) \bmod q$.

The Discrete Logarithm Problem and Assumption

- The discrete log problem
  - Choose *G,g* at random (from a certain family *G* of groups), where *G* is a cyclic group and *g* is a generator
  - Choose a random element $h \in G$
  - Give the adversary the input *(G,|G|,g,h)*
  - The adversary succeeds if it outputs $\log_g h$
- The discrete log assumption
  - There exists a family *G* of groups for which the discrete log problem is hard
    - Namely, the adversary has negligible success probability.

Cyclic groups of prime order

- (The order of a group G is the number of elements in the group)
- Z_p* has order p-1 (and p-1 is even and therefore non-prime).
- We will need to work in groups of *prime* order.

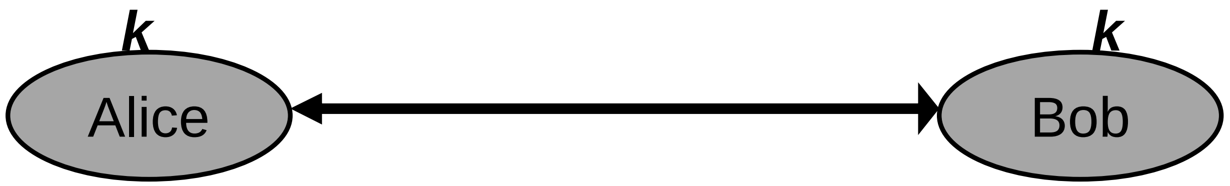
- If p=2q+1, and q is prime, then Z_p* has a subgroup of order q (namely, a subgroup of prime order).

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 where 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}.

Classical symmetric ciphers

- Alice and Bob share a private key *k.*
- System is secure as long as *k* is secret.
- Major problem: generating and distributing *k.*



Diffie and Hellman: “New Directions in Cryptography”, 1976.

- “We stand today on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing…

  …such applications create a need for new types of cryptographic systems which minimize the necessity of secure key distribution…

  …theoretical developments in information theory and computer science show promise of providing provably secure cryptosystems, changing this ancient art into a science.”

Diffie-Hellman

- Came up with the idea of public key cryptography

public key$_{Bob}$
secret key$_{Bob}$
Alice
Bob

Everyone can learn Bob's public key and encrypt messages to Bob.
Only Bob knows the decryption key and can decrypt.

Key distribution is greatly simplified.

- Diffie and Hellman did not have an implementation for a public key encryption system
- Suggested a method for key exchange over insecure communication lines, that is still in use today.

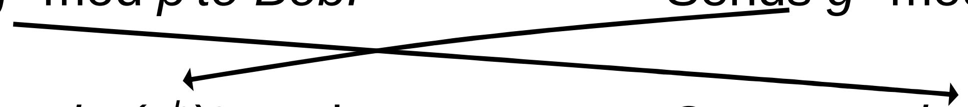
Public Key-Exchange

- 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.

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}.

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

- Why don't we implement Diffie-Hellman in *Zp**

  (but rather in a subgroup $H \subset Zp^*$, for $p=2q+1$, of order q, and a generator g of $H \subset Zp^*$).

- For the system to be secure, we need that the DDH assumption holds.
- This assumption does not hold in *Zp** (see discussion below)

Quadratic Residues

- The square root of $x \in Z_p^*$ is $y \in Z_p^*$ s.t. $y^2=x$ mod p.
- Examples: sqrt(2) mod 7 = 3, sqrt(3) mod 7 doesn't exist.
- How many square roots does $x \in Z_p^*$ have?
  - If a and b are square roots of x, then $x=a^2=b^2$ mod p.
  - Therefore for any two square roots of any number x it holds that $(a-b)(a+b)=0$ mod p.
  - Therefore either $a=b$ or $a= -b$ modulo p.
  - Therefore x has either 2 or 0 square roots, and is denoted as a Quadratic Residue (QR) or Non Quadratic Residue (NQR), respectively.
  - There are exactly $(p-1)/2$ QRs.

Quadratic Residues

- x$^{(p-1)/2}$ is either 1 or -1 in Z_p^* (since (x$^{(p-1)/2}$)^2 is always *1).*
- Euler's theorem: $x \in Z_p^*$ is a QR iff $x^{(p-1)/2} = 1$ mod *p.*
- *Legendre's symbol:*

$$\left(\frac{x}{p}\right) = \begin{cases} 1 & x \text{ is a QR in } Z_p^* \\ -1 & x \text{ is an NQR in } Z_p^* \\ 0 & x = 0 \bmod p \end{cases}$$

- Legendre's symbol can be efficiently computed as $x^{(p-1)/2}$ mod *p.*
- Another way to look at this: let g be a generator of Z_p^*. Then every *x* can be written as $x=g^i$ mod *p*. It holds that *x* is a QR iff *i* is even.

- The quadratic residues form a subgroup of order *(p-1)/2   (=q)*

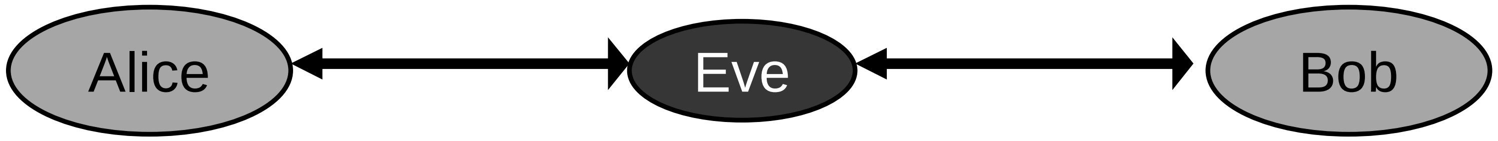
Does the DDH assumption hold in Z_p^*?

- The DDH assumption does not hold in Z_p^*
  - Assume that both $x=g^a$ and $y=g^b$ are QRs in Z_p^*.
  - Then g^{ab} is also a QR, whereas a random g^c is an NQR with probability ½.

- Solution: (work in a subgroup of prime order)
  - Set $p=2q+1$, where q is prime.
  - $\phi(Z_p^*) = p-1 = 2q$. Therefore Z_p^* has a subgroup H of *prime* order q.
  - Let g be a generator of H *(for example, g is a QR in Z_p^*).*
  - The DDH assumption is believed to hold in *H.* (The Legendre symbol is always 1.)

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 $\longrightarrow$

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

Eve changes g^b to g^d $\longleftarrow$

  - *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)

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^**.*

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, we can map each message $m \in \{1,\dots,(p-1)/2\}$ to the value m^2 mod *p*, which is in *H*.
  - 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)