Cyclic groups

- Let G be a finite group of order m (written multiplicatively)
- Let g be some element of G
- Consider the set $\langle g \rangle = \{g^0, g^1, \ldots\}$
  - We know $g^m = 1 = g^0$, so the set has $\leq m$ elements
  - If the set has m elements, then it is all of G!
    - In this case, we say g is a *generator* of G
    - If G has a generator, we say G is *cyclic*

令G为有限群，且$g \in G$，则g的order是使得$g^i = 1$的最小i.

生成元g能通过运算获得所有G中的元素.

Proposition 1: finite group G, $g \in G$ with order i, then for any integer x, we have

$$g^x = g^{x \bmod i}$$

Proposition 2: finite group G, $g \in G$ with order i, then for any integer x, y,

$$g^x = g^y \iff x = y \bmod i$$

任意群G的单位元是唯一的有order为1的元素，可生成群$\langle 1 \rangle = \{1\}$.

判断g是否为G的生成元，只需找是否有 $g^k = 1$ 且 $k \neq 0$. 若有，则g是生成元.

Proposition 3: finite group G of order m, if $g \in G$ has order i, then $i \mid m$.

∵G的order为m ∴$g^m = 1$.

令$m = iq + r$ $(0 \leq r < i)$，则有 $g^m = g^{iq+r} = (g^i)^q \cdot g^r = 1^q \cdot g^r = g^r = 1$

i) $r = 0$，符合

ii) $r \neq 0$，则与$\mathrm{ord}(g) = i$矛盾

故r必须为0，则 $i \mid m$

Example

$\mathbb{Z}_N^*$：不含0的$\mathbb{Z}_N$，且元素与N互质.

- $\mathbb{Z}_N$
  - Cyclic (for any N); 1 is always a generator: $\{0, 1, 2, \ldots, N-1\}$
- $\mathbb{Z}_8$
  - Is 3 a *generator*? $\{0, 3, 6, 1, 4, 7, 2, 5\}$ - Yes!
  - Is 2 a *generator*? $\{0, 2, 4, 6\}$ - No! 无法获得$\mathbb{Z}_8$中的所有元素
- $\mathbb{Z}_{11}^*$
  - Is 3 a *generator*? $\{1, 3, 9, 5, 4\}$ - No!
  - Is 2 a *generator*? $\{1, 2, 4, 8, 5, 10, 9, 7, 3, 6\}$ - Yes!
  - Is 8 a *generator*? $\{1, 8, 9, 6, 4, 10, 3, 2, 5, 7\}$ - Yes!

对于不是生成元的其他元素，其order也能整除G.

若 $a \in \mathbb{Z}_N$，只有 $\gcd(a, N) = 1$ 才可能是生成元

- **Theorem 11.1** Any group of prime order is cyclic, and every non-identity element is a generator.

质数大小的群是循环群，且非单位元的元素都是生成元。

- **Theorem 11.2** If p is prime, then $\mathbb{Z}_p^*$ is cyclic
  - **Note**: the order is $p - 1$, which is not prime for $p > 3$

证 Theorem 11.1：

由 proposition 3 可知，若 $m = p$ 为一个质数，则 $\forall g \in G$，有 $\text{ord}(g) = 1$ 或 p.（因为质数只有1和p两个因子）

只有单位元有 order 1，所以其他元素都有 order p 并能生成 G.

Uniform sampling

- Given cyclic group G of order q along with generator g, easy to sample a uniform $h \in G$
  - Choose uniform $x \in \{0, \ldots, q-1\}$: set $h := g^x$

- Fix *cyclic* group G of order q, and *generator* g

- We know that $\{g^0, g^1, \ldots, g^{q-1}\} = G$
  - For every $h \in G$, there is a unique $x \in \mathbb{Z}_q$, s.t. $g^x = h$
  - Define $\log_g h$ to be this x – the *discrete logarithm* of h with respect to g (in the group G)

找到离散对数是困难的，因为一般要全部计算出来

Examples

- In $\mathbb{Z}_{11}^*$
  - What is $\log_2 9$?
    - $\langle 2 \rangle = \{1, 2, 4, 8, 5, 10, 9, 7, 3, 6\}$, so $\log_2 9 = 6$
  - What is $\log_8 9$?
    - $\langle 8 \rangle = \{1, 8, 9, 6, 4, 10, 3, 2, 5, 7\}$, so $\log_8 9 = 2$

- In $\mathbb{Z}_{13}^*$
  - What is $\log_2 9$?
    - $\langle 2 \rangle = \{1, 2, 4, 8, 3, 6, 12, 11, 9, 5, 10, 7\}$, so $\log_2 9 = 8$

Discrete-logarithm problem (informal)

- DLog problem in G:
  Given *generator* g and element h, compute $\log_g h$
- DLog assumption in G:
  Solving the discrete log problem in G is hard
- In $\mathbb{Z}^*_{3092091139}$
  - What is $\log_2 1656755742$?

Discrete-logarithm problem

- Let $\mathcal{G}$ be a group-generation algorithm
  - On input 1^n, outputs a cyclic group G, its order q (with $\|q\| = n$), and a generator g
- For algorithm A, define *exp't* $Dlog_{A,\mathcal{G}}(n)$:
  - Compute $(G, q, g) \leftarrow \mathcal{G}(1^n)$
  - Choose uniform $h \in G$
  - Run $A(G, q, g, h)$ to get x
  - Experiment evaluates to 1 if $g^x = h$
- **Definition 11.3** The *discrete-logarithm problem* is hard relative to $\mathcal{G}$ if for all PPT algorithms A,
  $$\Pr[Dlog_{A,\mathcal{G}}(n) = 1] \leq negl(n)$$

安全参数n是q的二进制长度.

Diffie-Hellman problems

- Fix cyclic group G and *generator* g
- Define $DH_g(h_1, h_2) = DH_g(g^x, g^y) = g^{xy}$
- In $\mathbb{Z}^*_{11}$
  - $\langle 2 \rangle = \{1, 2, 4, 8, 5, 10, 9, 7, 3, 6\}$
  - So $DH_2(7, 5) = ?$ $DHg(2^7, 2^4) = g^{7\cdot 4} = 2^8 = 3$
- In $\mathbb{Z}^*_{3092091139}$
  - What is $DH_2(1656755742, 938640663) = ?$
  - Is 1994993011 the answer, or is it just a random element of $\mathbb{Z}^*_{3092091139}$?

Diffie-Hellman assumptions

- *Computational* Diffie-Hellman (CDH) problem:
  - Given g, h_1, h_2, compute $DH_g(h_1, h_2)$

计算出 $DHg(h_1, h_2)$

- *Decisional* Diffie-Hellman (DDH) problem:
  - Given g, h_1, h_2, distinguish $DH_g(h_1, h_2)$ from a uniform element of G

分辨 $DHg(h_1, h_2)$ 与 G中的元素.

DDH problem

- Let $\mathcal{G}$ be a group-generation algorithm
  - On input 1^n, outputs a cyclic group G, its order q (with $\|q\| = n$), and a generator g

- The DDH problem is hard relative to $\mathcal{G}$ if for all PPT algorithm A:

$$|\Pr[A(G, q, g, g^x, g^y, g^z) = 1] - \Pr[A(G, q, g, g^x, g^y, g^{xy}) = 1]| \le \epsilon(n)$$

h_1 h_2 随机取 (h_1, G中)　　h_1 h_2 DHg(h_1, h_2)

Relating the Diffie-Hellman problems

- Relative to $\mathcal{G}$
  - If the discrete-logarithm problem is easy, so is the CDH problem
  - If the CDH problem is easy, so is the DDH problem
  - I.e., the DDH assumption is *stronger* than the CDH assumption
  - I.e., the CDH assumption is *stronger* than the dlog assumption

dlog easy ⇒ CDH easy ⇒ DDH easy

但反过来不一定，即 DDH easy ⇏ dlog easy / CDH easy

Group selection

- The *discrete logarithm* is not hard in all groups!
- Nevertheless, there are certain groups where the problem is believed to be hard
- For cryptographic applications, best to use *prime-order* groups
  - The *dlog* problem becomes easier if the order of the group has small prime factors
  - Prime-order groups have several nice features: e.g., every element except identity is a generator
- Two common choices of groups

离散对数问题不是对全部群都困难

最好用order为质数的群（因为这样的群必为循环群）

若群的order有小质因子，则dlog问题会简单

Choice 1

- *Prime-order* subgroup of $\mathbb{Z}_p^*$, p prime
  - E.g., $p = tq + 1$ for q prime
  - Take the subgroup of t^{th} powers, i.e., $G = \{[x^t \bmod p] | x \in \mathbb{Z}_p^*\}$
  - This is a group
  - It has order $(p-1)/t = q$
  - Since q is prime, the group must be *cyclic*
  - Generalizations based on finite fields are also used

Choice 2

- *Prime-order* subgroup of an *elliptic curve* group
  - See book for details

- We will describe algorithm in "abstract" groups
  - Can ignore details of the underlying group in the analysis
  - Can instantiate with any (appropriate) group for an implementation

Concrete parameters

- We have discussed two classes of cryptographic assumptions
  - *Factoring-based* (factoring, RSA assumptions)
  - *Dlog-based* (dlog, CDH, and DDH assumptions)

- All these problems are (believed to be) "hard", i.e., to have no polynomial-time algorithms
  - But how hard are they, concretely?

- The goal here is to give an idea as to how parameters are calculated, and what relevant parameters are

Security

- Recall: For symmetric-key algorithms
  - Block cipher with n-bit key $\approx$ security againt 2^n-time attacks
  - Hash functions with n-bit output $\approx$ security againt $2^{n/2}$-time attacks

- Factoring of a modulus of size 2^n (i.e., length n) using exhaustive search takes $2^{n/2}$ time

- Computing discrete logarithms in a group of order 2^n takes 2^n time
  - Are these the best algorithms possible?

Algorithm for factoring

- There exist algorithms factoring an integer N that run in much less than $2^{\|N\|/2}$ time

- Best known algorithm (asymptotically):
  general number field sieve
  - Running time (heuristic): $2^{O(\|N\|^{1/3}\log^{2/3}\|N\|)}$
  - Makes a huge difference in practice
  - Exact constant term also important!

筛法

Algorithm for dlog

- Two classes of algorithms:
  - Ones that work for *arbitrary* ("generic") groups
  - Ones that target specific groups
    - Recall that in some groups the problem is not even hard

- Best "generic" algorithms:
  - Time $2^{n/2}$ in a group of order $\approx 2^n$
  - This is known to be optimal for generic algorithms

- Best known algorithm for (subgroups of) $\mathbb{Z}_p^*$:
  number field sieve
  - Running time (heuristic): $2^{O(\|p\|^{1/3}\log^{2/3}\|p\|)}$

- For (appropriately chosen) elliptic-curve groups, nothing better than generic algorithms is known!
  - This is why elliptic-curve groups can allow for more-efficient cryptography

Choosing parameters

- As recommended by NIST (112-bit security):
  - *Factoring*: 2048-bit modulus
  - *Dlog*: order-q subgroup of $\mathbb{Z}_p^*$: $\|q\| = 224$, $\|p\| = 2024$
  - *Dlog*, elliptic-curve group of order q: $\|q\| = 224$

- Much longer than for symmetric-key algorithms!
  - Explains in part why public-key crypto is less efficient than symmetric-key crypto

Private-key cryptography

- *Private-key cryptography* allows two users who *share a secret key* to establish a "secure channel"
- The need to share a *secret key* has several drawbacks
- How do users share a key in the first place?
  - Need to share the key using a secure channel
- This problem can be solved in some settings
  - E.g., physical proximity, trusted courier, ...
  - **Note**: this does not make *private-key cryptography* useless!
- Can be difficult or expensive to solve in other settings

The key-management problem

- Imagine an organization with N employees, where each pair of employees might need to communicate securely
- Solution using *private-key cryptography*
  - Each user shares a key with all other users
  - $\Rightarrow$ Each user must store/manage $N-1$ secret keys!
  - $\Rightarrow$ $O(N^2)$ keys overall!

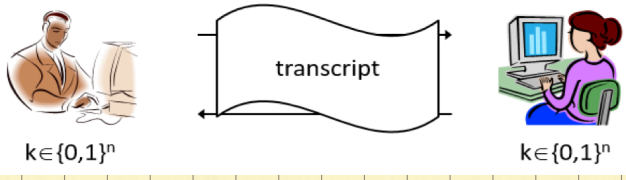
需要保存其他所有用户的密钥

New directions

- Main ideas:
  - Some problems exhibit *asymmetry* - easy to compute, but hard to invert (*factoring*, *RSA*, *group exponentiation*, ...)
  - Use this *asymmetry* to enable two parties to agree on a shared secret key via public discussion
    - *Key exchange*

Key exchange

- *Security goal*: even after observing the transcript, the shared key k should be *indistinguishable* from a uniform key



窃听者无法将 shared key k 与随机数区分开来

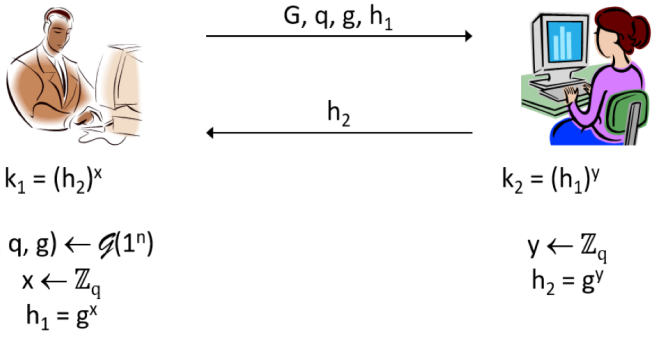
- Fix a key-exchange protocol Π and an attacker (passive eavesdropper) A
- Define the following experiment $KE_{A,\Pi}(n)$:
  - Honest parties run Π using security parameter n, resulting in a transcript *trans* and (shared) key k
  - Choose uniform bit b. If $b=0$, then set $k'=k$; if $b=1$, then choose uniform $k' \in \{0,1\}^n$
  - Give *trans* and k' to A, which outputs a bit b'
  - Exp't evaluates to 1 (A *succeeds*) if $b=b'$
- **Definition 12.1** Key-exchange protocol Π is *secure* (against passive eavesdropping) if for *all* PPT A it holds that

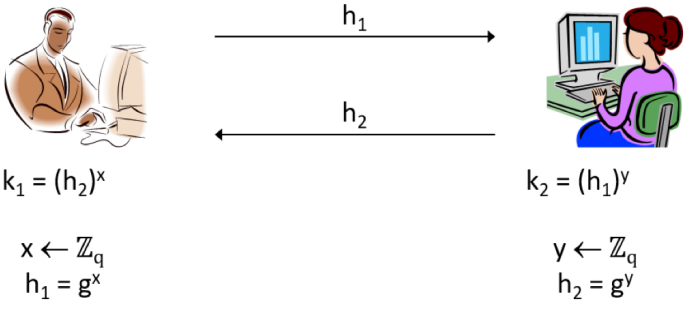
$$\Pr[KE_{A,\Pi}(n)=1] \le 1/2 + negl(n)$$

- Being unable to compute the key given the transcript is not a strong enough guarantee

- Indistinguishability of the shared key from uniform is a much stronger guarantee
  - And is necessary if the shared key will subsequently be used for *private-key crypto*!

Diffie-Hellman key exchange





Security

- Eavesdropper sees G, q, g, g^x, g^y
- Shared key k is g^{xy}
- Computing k from the transcript is exactly the *computational* Diffie-Hellman problem
- Distinguishing k from uniform group element is exactly the *decisional* Diffie-Hellman problem
  $\Rightarrow$ If the DDH problem is hard relative to $\mathcal{G}$, this is a *secure* key-exchange protocol!

A subtlety

- We wanted our key-exchange protocol to give us a uniform(-looking) key $k \in \{0,1\}^n$
- Instead we have a uniform(-looking) group element $k \in G$
  - Not clear how to use this as, e.g., an AES key
- Solution: *key derivation*
  - Set $k' = H(k)$ for suitable hash function H

$h_1 = g^x$, $h_2 = g^y$

任意一方得到 h_1 或 h_2 后可算出 g^{xy}.
如已知 $h_1 = g^x$，获得 $h_2 = g^y$ 后可得 $h_1 h_2 = g^x g^y = g^{x+y}$
(在模意义下)

现实中 G, q, g 一般公开，仅交换 h_1, h_2

窃听者知道 $k = g^{xy}$，但无法得知 x, y 具体多少