Lecture 8 – Public-Key Encryption and Computational Number Theory

COSC-466 Applied Cryptography

Adam O'Neill

Adapted from

http://cseweb.ucsd.edu/~mihir/cse107/

Recall Symmetric-Key Crypto

- In this setting, if Alice wants to communicate secure with Bob they need a shared key K_{AB}.

Recall Symmetric-Key Crypto

- In this setting, if Alice wants to communicate secure with Bob they need a shared key K_{AB}.
- If Alice wants to also communicate with Charlie they need a shared key K_{AC}.

Recall Symmetric-Key Crypto

- In this setting, if Alice wants to communicate secure with Bob they need a shared key K_{AB}.
- If Alice wants to also communicate with Charlie they need a shared key K_{AC}.
- If Alice generates K_{AB} and K_{AC} they must be communicated to Bob and Charlie over secure channels. How can this be done?

Public-Key Crypto

- Alice has a secret key that is shared with nobody, and a public key that is shared with everybody.

Public-Key Crypto

- Alice has a secret key that is shared with nobody, and a public key that is shared with everybody.
- Anyone can use Alice's public key to send her a private message.

Public-Key Crypto

- Alice has a secret key that is shared with nobody, and a public key that is shared with everybody.
- Anyone can use Alice's public key to send her a private message.
- Public key is like a phone number: Anyone can look it up in a phone book.

Syntax and Correctness of PKE

A public-key (or asymmetric) encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consists of three algorithms, where

- $\mathcal{K}$ is the key-generation algorithm that on input 1^k outputs a private (or secret) key sk and public key pk.

$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$

k is a security parameter (think of key length)

- $\mathcal{E}$ is the encryption algorithm that on inputs pk and m outputs a ciphertext c

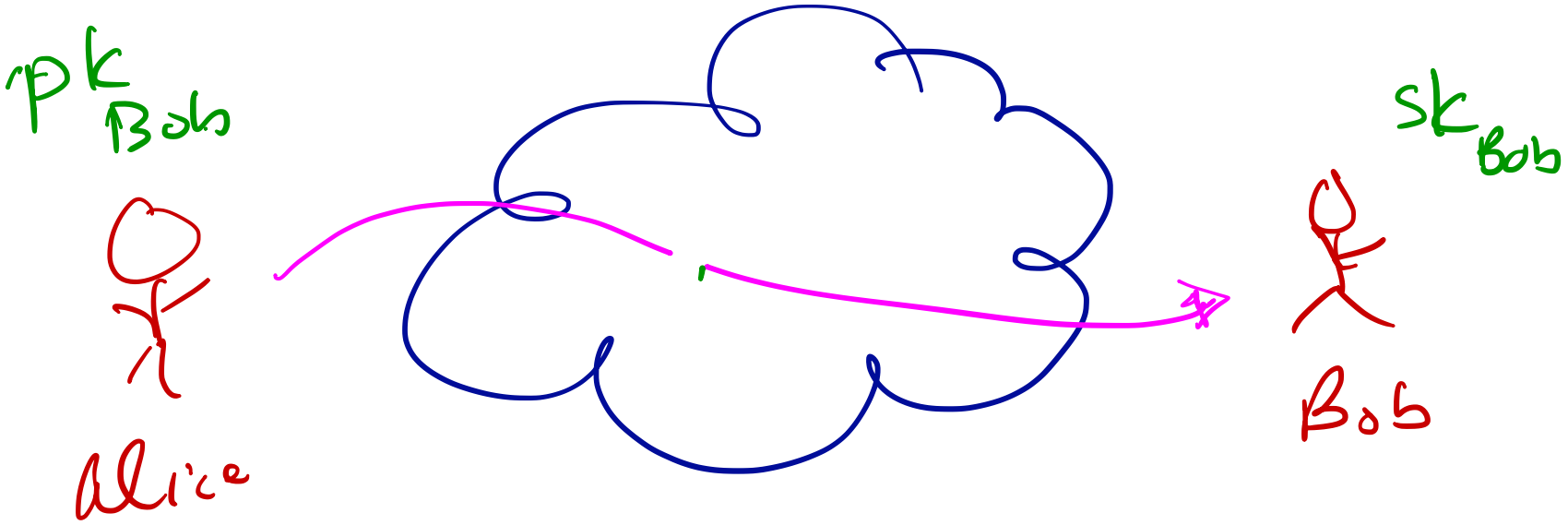
pk: public key; m: message

$c \xleftarrow{\$} \mathcal{E}_{pk}(m)$

- $\mathcal{D}$ is the decryption algorithm that on inputs sk and c outputs m or $\perp$

~~secret key~~ sk; c: ciphertext

Intended Usage



Code Obfuscation Perspective

IND-CPA

Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a PKE scheme and A an adversary.

Game $\mathrm{Left}_{\mathcal{AE}}$	Game $\mathrm{Right}_{\mathcal{AE}}$
procedure Initialize	**procedure Initialize**
$(pk, sk) \xleftarrow{\$} \mathcal{K}$; return pk	$(pk, sk) \xleftarrow{\$} \mathcal{K}$; return pk
procedure LR(M_0, M_1)	**procedure LR**(M_0, M_1)
Return $C \xleftarrow{\$} \mathcal{E}_{pk}(M_0)$	Return $C \xleftarrow{\$} \mathcal{E}_{pk}(M_1)$

Associated to $\mathcal{AE}, A$ are the probabilities

$$\Pr\left[\mathrm{Left}^{A}_{\mathcal{AE}} \Rightarrow 1\right] \qquad \Pr\left[\mathrm{Right}^{A}_{\mathcal{AE}} \Rightarrow 1\right]$$

that A outputs 1 in each world. The ind-cpa advantage of A is

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{AE}}(A) = \Pr\left[\mathrm{Right}^{A}_{\mathcal{AE}} \Rightarrow 1\right] - \Pr\left[\mathrm{Left}^{A}_{\mathcal{AE}} \Rightarrow 1\right]$$

A More Basic Problem: Key Exchange

How to Build Key Exchange?

- We need to take a trip into computational number theory….

Hardy, in his essay A Mathematician's Apology writes:

"Both Gauss and lesser mathematicians may be justified in rejoicing that there is one such science [number theory] at any rate, and that their own, whose very remoteness from ordinary human activities should keep it gentle and clean"

Notation

$\mathbf{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ $\mathbb{Z}$ $\mathbb{N}$ $\mathbb{Z}_+$

$\mathbf{N} = \{0, 1, 2, \ldots\}$

$\mathbf{Z}_+ = \{1, 2, 3, \ldots\}$

For $a, N \in \mathbf{Z}$ let $\gcd(a, N)$ be the largest $d \in \mathbf{Z}_+$ such that d divides both a and N.

Example: $\gcd(30, 70) = 10$.

a is relatively prime to b if $\gcd(a, b) = 1$.

Integers mod N

For $N \in \mathbf{Z}_+$, let

- $\mathbf{Z}_N = \{0, 1, \ldots, N-1\}$
- $\mathbf{Z}_N^* = \{a \in \mathbf{Z}_N \ : \ \gcd(a, N) = 1\}$
- $\varphi(N) = |\mathbf{Z}_N^*|$

$\varphi(N)$ = # relatively prime to N from 1 to $N-1$

Division and MOD

$\text{INT-DIV}(a, N)$ returns (q, r) such that

- $a = qN + r$
- $0 \leq r < N$

Refer to q as the quotient and r as the remainder. Then

$$a \bmod N = r \in \mathbf{Z}_N$$

is the remainder when a is divided by N.

Groups

Let G be a non-empty set, and let $\cdot$ be a binary operation on G. This means that for every two points $a, b \in G$, a value $a \cdot b$ is defined.

Example: $G = \mathbf{Z}_{12}^*$ and "$\cdot$" is multiplication modulo 12, meaning

$$a \cdot b = ab \bmod 12$$

Def: We say that G is a *group* if it has four properties called closure, associativity, identity and inverse that we present next.

Fact: If $N \in \mathbf{Z}_+$ then $G = \mathbf{Z}_N^*$ with $a \cdot b = ab \bmod N$ is a group.

Groups: Closure

Closure: For every $a, b \in G$ we have $a \cdot b$ is also in G.

Example: $G = \mathbf{Z}_{12}$ with $a \cdot b = ab$ does not have closure because $7 \cdot 5 = 35 \notin \mathbf{Z}_{12}$.

Fact: If $N \in \mathbf{Z}_+$ then $G = \mathbf{Z}_N^*$ with $a \cdot b = ab \bmod N$ satisfies closure, meaning

$$\gcd(a, N) = \gcd(b, N) = 1 \text{ implies } \gcd(ab \bmod N, N) = 1$$

Example: Let $G = \mathbf{Z}_{12}^* = \{1, 5, 7, 11\}$. Then

$$5 \cdot 7 \bmod 12 = 35 \bmod 12 = 11 \in \mathbf{Z}_{12}^*$$

Exercise: Prove the above Fact.

Groups: Associativity

Associativity: For every $a, b, c \in G$ we have $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.

Fact: If $N \in \mathbf{Z}_+$ then $G = \mathbf{Z}_N^*$ with $a \cdot b = ab \bmod N$ satisfies associativity, meaning

$$((ab \bmod N)c) \bmod N = (a(bc \bmod N)) \bmod N$$

Example:

$$\begin{aligned}(5 \cdot 7 \bmod 12) \cdot 11 \bmod 12 &= (35 \bmod 12) \cdot 11 \bmod 12 \\ &= 11 \cdot 11 \bmod 12 = 1\end{aligned}$$

$$\begin{aligned}5 \cdot (7 \cdot 11 \bmod 12) \bmod 12 &= 5 \cdot (77 \bmod 12) \bmod 12 \\ &= 5 \cdot 5 \bmod 12 = 1\end{aligned}$$

Exercise: Given an example of a set G and a natural operation $a, b \mapsto a \cdot b$ on G that satisfies closure but *not* associativity.

Groups: Identity Element

Identity element: There exists an element $\mathbf{1} \in G$ such that $a \cdot \mathbf{1} = \mathbf{1} \cdot a = a$ for all $a \in G$.

Fact: If $N \in \mathbf{Z}_+$ and $G = \mathbf{Z}_N^*$ with $a \cdot b = ab \bmod N$ then 1 is the identity element because $a \cdot 1 \bmod N = 1 \cdot a \bmod N = a$ for all a.

Groups: Inverses

Inverses: For every $a \in G$ there exists a unique $b \in G$ such that $a \cdot b = b \cdot a = \mathbf{1}$.

This b is called the inverse of a and is denoted a^{-1} if G is understood.

Fact: If $N \in \mathbf{Z}_+$ and $G = \mathbf{Z}_N^*$ with $a \cdot b = ab \bmod N$ then $\forall a \in \mathbf{Z}_N^* \quad \exists b \in \mathbf{Z}_N^*$ such that $a \cdot b \bmod N = 1$.

We denote this unique inverse b by $a^{-1} \bmod N$.

Example: $5^{-1} \bmod 12$ is the $b \in \mathbf{Z}_{12}^*$ satisfying $5b \bmod 12 = 1$, so $b =$

Computational Shortcuts

What is $5 \cdot 8 \cdot 10 \cdot 16 \bmod 21$?

Slow way: First compute

$$5 \cdot 8 \cdot 10 \cdot 16 = 40 \cdot 10 \cdot 16 = 400 \cdot 16 = 6400$$

and then compute $6400 \bmod 21 = 16$

Fast way:

$5 \cdot 8 \bmod 21 = 16$

$16 \cdot 10 \bmod 21 \ldots$

Exponentiation

Let G be a group and $a \in G$. We let $a^0 = \mathbf{1}$ be the identity element and for $n \geq 1$, we let

$$a^n = \underbrace{a \cdot a \cdots a}_{n}.$$

Also we let

$$a^{-n} = \underbrace{a^{-1} \cdot a^{-1} \cdots a^{-1}}_{n}.$$

This ensures that for all $i, j \in \mathbf{Z}$,

- $a^{i+j} = a^i \cdot a^j$
- $a^{ij} = (a^i)^j = (a^j)^i$
- $a^{-i} = (a^i)^{-1} = (a^{-1})^i$

Meaning we can manipulate exponents "as usual".

Examples

Let $N = 14$ and $G = \mathbf{Z}_N^*$. Then modulo N we have

$$5^3 = 125 \bmod 14$$
$$= 13 \bmod 14$$

and

$$5^{-3} = 13 \bmod 14$$

Group Orders

The order of a group G is its size $|G|$, meaning the number of elements in it.

Example: The order of $\mathbf{Z}_{21}^*$ is 12 because

$$\mathbf{Z}_{21}^* = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$$

$|\mathbb{Z}_{21}^*| = \varphi(21)$

Fact: Let G be a group of order m and $a \in G$. Then, $a^m = \mathbf{1}$.

Examples: Modulo 21 we have

- $5^{12} \equiv (5^3)^4 \equiv 20^4 \equiv (-1)^4 \equiv 1$
- $8^{12} \equiv (8^2)^6 \equiv (1)^6 \equiv 1$

Simplifying Exponentiation

Fact: Let G be a group of order m and $a \in G$. Then, $a^m = \mathbf{1}$.

Corollary: Let G be a group of order m and $a \in G$. Then for any $i \in \mathbf{Z}$,

$$a^i = a^{i \bmod m}.$$

Proof: Let $(q, r) \leftarrow \text{INT-DIV}(i, m)$, so that $i = mq + r$ and $r = i \bmod m$. Then

$$a^i = a^{mq+r} = (a^m)^q \cdot a^r$$

But $a^m = \mathbf{1}$ by Fact.

Corollary and Example

Corollary: Let G be a group of order m and $a \in G$. Then for any $i \in \mathbf{Z}$,

$$a^i = a^{i \bmod m}.$$

Example: What is 5^{74} mod 21?

$74 \bmod 2 = 2$

$5^2 \bmod 21 = 4.$

$G \quad a \in G$

$a^{|G|} = 1$

$a^0, a^1, \ldots, \ldots, 1$

Algorithms & Running-Time

In an algorithms course, the cost of arithmetic is often assumed to be $\mathcal{O}(1)$, because numbers are small. In cryptography numbers are

very, very BIG!

Typical sizes are 2^{512}, 2^{1024}, 2^{2048}.

Numbers are provided to algorithms in binary. The length of a, denoted $|a|$, is the number of bits in the binary encoding of a.

Example: $|7| = 3$ because 7 is 111 in binary.

Running time is measured as a function of the lengths of the inputs.

(a,b)

$aa' + bb' = (a,b)$ for some a', b'

quotient and remainder

Algorithm	Input	Output	Time
ADD	a, b	$a + b$	linear
MULT	a, b	ab	quadratic
INT-DIV	a, N	q, r	quadratic
MOD	a, N	$a \bmod N$	quadratic
EXT-GCD	a, N	(d, a', N')	quadratic
MOD-INV	$a \in \mathbf{Z}_N^*$, N	$a^{-1} \bmod N$	quadratic
MOD-EXP	a, n, N	$a^n \bmod N$	cubic
EXP_G	a, n	$a^n \in G$	$\mathcal{O}(\lvert n \rvert)$ G-ops

Extended GCD

$\mathrm{EXT\text{-}GCD}(a, N) \mapsto (d, a', N')$ such that

$$d = \gcd(a, N) = a \cdot a' + N \cdot N' \ .$$

Example: $\mathrm{EXT\text{-}GCD}(12, 20) = (4, 2, -3)$ because

$$4 = \gcd(12, 20) = 12 \cdot (-3) + 20 \cdot 2 \ .$$

EXT-GCD Algorithm

$\text{EXT-GCD}(a, N) \mapsto (d, a', N')$ such that

$$d = \gcd(a, N) = a \cdot a' + N \cdot N' \ .$$

Lemma: Let $(q, r) = \text{INT-DIV}(a, N)$. Then, $\gcd(a, N) = \gcd(N, r)$

Alg $\text{EXT-GCD}(a, N)$ // $(a, N) \neq (0, 0)$
if $N = 0$ then return $(a, 1, 0)$
else
  $(q, r) \leftarrow \text{INT-DIV}(a, N)$; $(d, x, y) \leftarrow \text{EXT-GCD}(N, r)$
  $a' \leftarrow y$; $N' \leftarrow x - qy$
  return (d, a', N')

Running time analysis is non-trivial (worst case is Fibonacci numbers) and shows that the time is $\mathcal{O}(|a| \cdot |N|)$. So the extended gcd can be computed in quadratic time.

Modular Inverse

For a, N such that $\gcd(a, N) = 1$, we want to compute $a^{-1} \bmod N$, meaning the unique $a' \in \mathbf{Z}_N^*$ satisfying $aa' \equiv 1 \pmod{N}$.

But if we let $(d, a', N') \leftarrow \text{EXT-GCD}(a, N)$ then

$$d = 1 = \gcd(a, N) = a \cdot a' + N \cdot N'$$

But $N \cdot N' \equiv 0 \pmod{N}$ so $aa' \equiv 1 \pmod{N}$

Alg $\text{MOD-INV}(a, N)$
$(d, a', N') \leftarrow \text{EXT-GCD}(a, N)$
return $a' \bmod N$

Modular inverse can be computed in quadratic time.

Exponentiation

Let G be a group and $a \in G$. For $n \in \mathbf{N}$, we want to compute $a^n \in G$.

We know that

$$a^n = \underbrace{a \cdot a \cdots a}_{n}$$

Consider:

$y \leftarrow 1$
for $i = 1, \ldots, n$ do $y \leftarrow y \cdot a$
return y

a^n if $|n|$ is large

Question: Is this a good algorithm?

Square-and-Multiply

Let $\text{bin}(n) = b_{k-1} \dots b_0$ be the binary representation of n, meaning

$$n = \sum_{i=0}^{k-1} b_i 2^i$$

Alg $\text{EXP}_G(a, n)$ // $a \in G$, $n \geq 1$
$b_{k-1} \dots b_0 \leftarrow \text{bin}(n)$
$y \leftarrow 1$
for $i = k-1$ downto 0 do $y \leftarrow y^2 \cdot a^{b_i}$
return y

The running time is $\mathcal{O}(|n|)$ group operations.

MOD-EXP(a, n, N) returns a^n mod N in time $\mathcal{O}(|n| \cdot |N|^2)$, meaning is cubic time.

Generators and Cyclic Groups

Let G be a group of order m and let $g \in G$. We let

$$\langle g \rangle = \{ g^i \ : \ i \in \mathbf{Z} \} .$$

Fact: $\langle g \rangle = \{ g^i \ : \ i \in \mathbf{Z}_m \}$

Exercise: Prove the above Fact.

Fact: The size $|\langle g \rangle|$ of the set $\langle g \rangle$ is a divisor of m

Note: $|\langle g \rangle|$ need not equal m!

Definition: $g \in G$ is a generator (or primitive element) of G if $\langle g \rangle = G$, meaning $|\langle g \rangle| = m$.

Definition: G is cyclic if it has a generator, meaning there exists $g \in G$ such that g is a generator of G.

Example

Let $G = \mathbf{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, which has order $m = 10$.

i	0	1	2	3	4	5	6	7	8	9	10
$2^i \mod 11$	1	2	4	8	5	10	9	7	3	6	1
$5^i \mod 11$	1	5	3	4	9	1	5	3	4	9	1

so

$$\begin{aligned} \langle 2 \rangle &= \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \\ \langle 5 \rangle &= \{1, 3, 4, 5, 9\} \end{aligned}$$

- 2 a generator because $\langle 2 \rangle = \mathbf{Z}_{11}^*$.
- 5 is not a generator because $\langle 5 \rangle \neq \mathbf{Z}_{11}^*$.
- $\mathbf{Z}_{11}^*$ is cyclic because it has a generator.

Discrete Logarithms

If $G = \langle g \rangle$ is a cyclic group of order m then for every $a \in G$ there is a unique exponent $i \in \mathbf{Z}_m$ such that $g^i = a$. We call i the discrete logarithm of a to base g and denote it by

$$\mathrm{DLog}_{G,g}(a)$$

The discrete log function is the inverse of the exponentiation function:

$$\begin{aligned} \mathrm{DLog}_{G,g}(g^i) &= i \quad \text{for all } i \in \mathbf{Z}_m \\ g^{\mathrm{DLog}_{G,g}(a)} &= a \quad \text{for all } a \in G. \end{aligned}$$

$\mathrm{Dlog}_g(a)$

Example

Let $G = \mathbf{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, which is a cyclic group of order $m = 10$. We know that 2 is a generator, so $\mathrm{DLog}_{G,2}(a)$ is the exponent $i \in \mathbf{Z}_{10}$ such that $2^i \bmod 11 = a$.

i	0	1	2	3	4	5	6	7	8	9
$2^i \bmod 11$	1	2	4	8	5	10	9	7	3	6

a	1	2	3	4	5	6	7	8	9	10
$\mathrm{DLog}_{G,2}(a)$	0	1	8	2	4	9	7	3	6	5

Finding Cyclic Groups

Fact 1: Let p be a prime. Then $\mathbf{Z}_p^*$ is cyclic.

Fact 2: Let G be any group whose order $m = |G|$ is a prime number. Then G is cyclic.

Note: $|\mathbf{Z}_p^*| = p - 1$ is not prime, so Fact 2 doesn't imply Fact 1!

Computing Discrete Logs

Let $G = \langle g \rangle$ be a cyclic group of order m with generator $g \in G$.

Input: $X \in G$

Desired Output: $\mathrm{DLog}_{G,g}(X)$

Current Best Algorithms

Group	Time to find discrete logarithms
$\mathbf{Z}_p^*$	$e^{1.92(\ln p)^{1/3}(\ln \ln p)^{2/3}}$
EC_p	$\sqrt{p} = e^{\ln(p)/2}$

"index calculus"

baby step giant step
pollard-rho

Here p is a prime and EC_p represents an elliptic curve group of order p.

Note: In the first case the actual running time is $e^{1.92(\ln q)^{1/3}(\ln \ln q)^{2/3}}$ where q is the largest prime factor of $p-1$.

In neither case is a polynomial-time algorithm known.

This (apparent, conjectured) computational intractability of the discrete log problem makes it the basis for cryptographic schemes in which breaking the scheme requires discrete log computation.

Current Records

In $\mathbf{Z}_p^*$:

$\lvert p \rvert$ **in bits**	**When**
431	2005
530	2007
596	2014

For elliptic curves, current record seems to be for $\lvert p \rvert$ around 113.

Why ECC?

Say we want 80-bits of security, meaning discrete log computation by the best known algorithm should take time 2^{80}. Then

- If we work in $\mathbf{Z}_p^*$ (p a prime) we need to set $|\mathbf{Z}_p^*| = p - 1 \approx 2^{1024}$
- But if we work on an elliptic curve group of prime order p then it suffices to set $p \approx 2^{160}$.

Why? Because

$$e^{1.92(\ln 2^{1024})^{1/3}(\ln \ln 2^{1024})^{2/3}} \approx \sqrt{2^{160}} = 2^{80}$$

1024 bit elements

160 bit elements

DL Problem

Let $G = \langle g \rangle$ be a cyclic group of order m, and A an adversary.

Game $\mathrm{DL}_{G,g}$

procedure Initialize
$x \xleftarrow{\$} \mathbf{Z}_m$; $X \leftarrow g^x$
return X

procedure Finalize(x')
return $(x = x')$

The dl-advantage of A is

$$\mathbf{Adv}^{\mathrm{dl}}_{G,g}(A) = \Pr\left[\mathrm{DL}^{A}_{G,g} \Rightarrow \mathsf{true}\right]$$

CDH Problem

Computational Diffie-Hellman

Let $G = \langle g \rangle$ be a cyclic group of order m, and A an adversary.

Game $\mathrm{CDH}_{G,g}$

procedure Initialize
$x, y \xleftarrow{\$} \mathbf{Z}_m$
$X \leftarrow g^x$; $Y \leftarrow g^y$
return X, Y

procedure Finalize(Z)
return $(Z = g^{xy})$

The cdh-advantage of A is

$$\mathbf{Adv}^{\mathrm{cdh}}_{G,g}(A) = \Pr\left[\mathrm{CDH}^{A}_{G,g} \Rightarrow \mathrm{true}\right]$$

Building Cyclic Groups

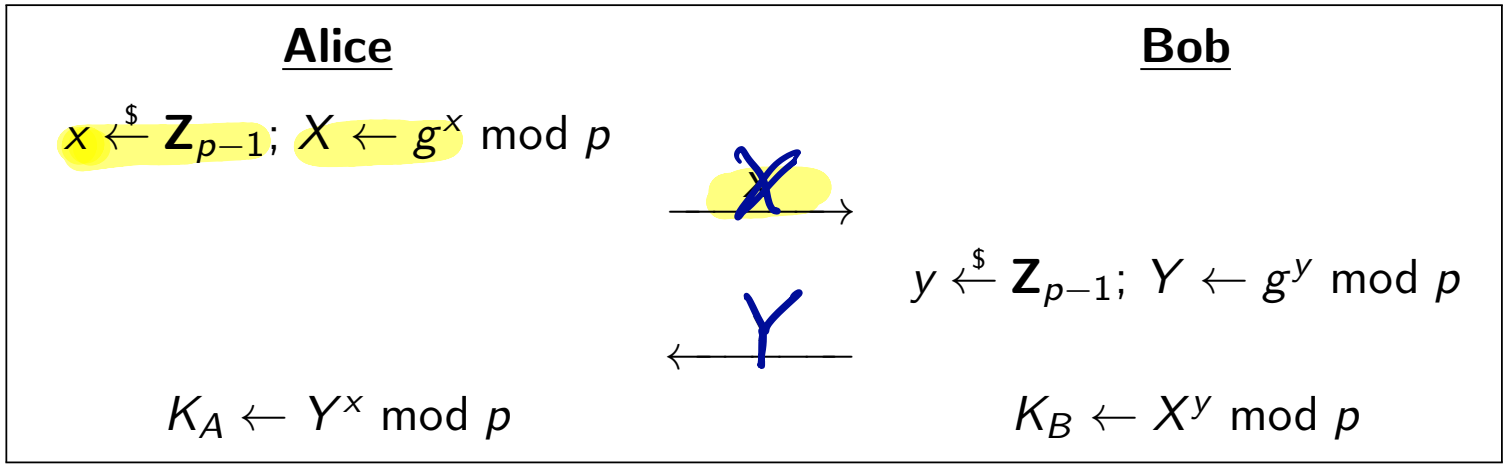
Choose random p of bit-length k until p is prime

// can test primality efficiently

Z_p^* cyclic group under multiplication

Diffie-Hellman Key Exchange

The following are assumed to be public: A large prime p and a generator g of $\mathbf{Z}_p^*$.



- $Y^x = (g^y)^x = g^{xy} = (g^x)^y = X^y$ modulo p, so $K_A = K_B$
- Adversary is faced with the CDH problem.

Weak form of key-agreement
- adversary is passive
- we really want key indistinguishable from $