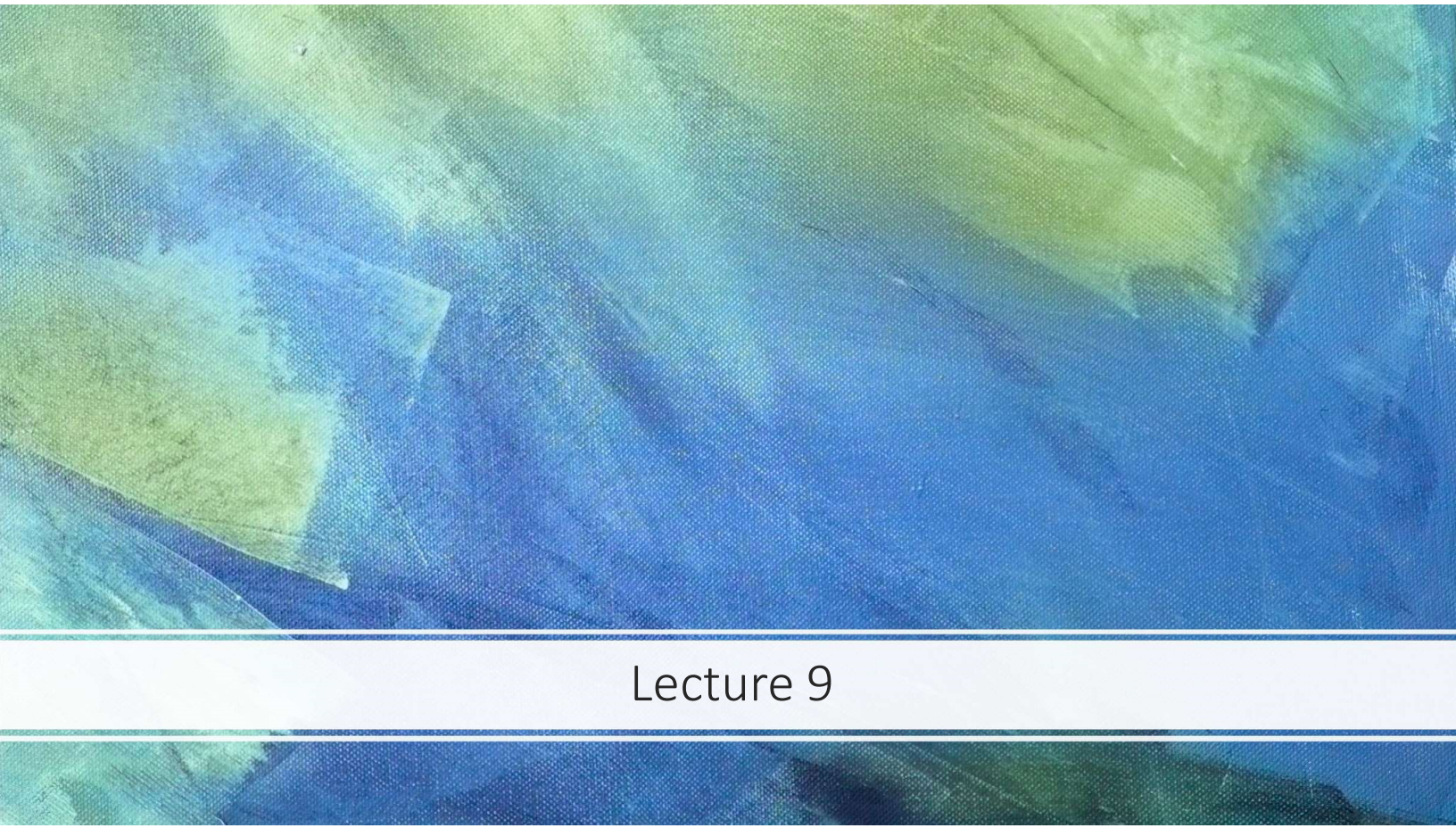
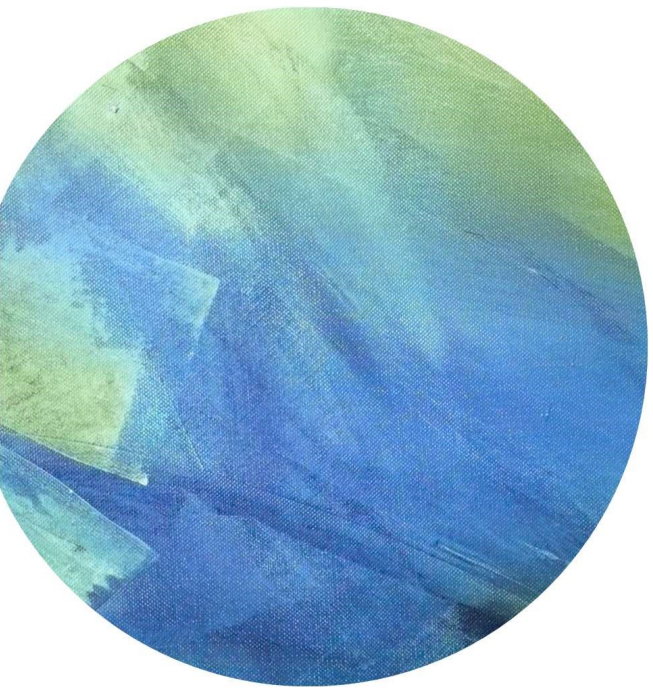

Lecture 9



Outline

Number Theory

Hard Problems

Key Exchange

Number Theory

Number theory: Recall

N denotes an n-bit positive integer. p denotes a prime.

- $\mathbb{Z}_N$ = { 0, 1, …, N-1 }

- $(\mathbb{Z}_N)^*$ = (set of invertible elements in Z_N) =
  = $\{ x \in Z_N : \gcd(x,N) = 1 \}$

Can find inverses efficiently using Euclid algorithm: time = $O(n^2)$

$\log_2 N$

Fermat's theorem (1640)

Thm: Let p be a prime

$$\forall\, x \in (Z_p)^* : \quad x^{p-1} = 1 \text{ in } Z_p^*$$

$\{0, 1, \ldots, p-1\}$

Example: p=5. $3^4 = 81 = 1$ in Z_5 $2^4 = 16 = 1$ in Z_5

So: $x \in (Z_p)^* \Rightarrow x \cdot x^{p-2} = 1 \Rightarrow x^{-1} = x^{p-2}$ in Z_p

another way to compute inverses, but less efficient than Euclid

Application: generating random primes

Suppose we want to generate a large random prime

say, prime p of length 1024 bits (i.e. $p \approx 2^{1024}$)

$n \nearrow$

Set p = random sequence of 1024 bits

Step 1: choose a random integer $p \in [\ 2^{1024}\ ,\ 2^{1025}-1\ ]$

Step 2: test if $2^{p-1} = 1$ in Z_p

If so, output p and stop. If not, goto step 1 .

Simple algorithm (not the best). **Pr[p not prime] < 2^{-60}**

The structure of $(Z_p)^*$

Thm (Euler): $(Z_p)^*$ is a **cyclic group**, that is

$\exists\, g \in (Z_p)^*$ such that $\{1, g, g^2, g^3, \ldots, g^{p-2}\} = (Z_p)^*$

g is called a **generator** of $(Z_p)^*$ $\{1, g, g^2 \ldots\}$ is exactly equal to Z_p^*.

Example: p=7. $\{1, 3, 3^2, 3^3, 3^4, 3^5\} = \{1, 3, 2, 6, 4, 5\} = (Z_7)^*$

$\{1, 2, 3, 4, 5, 6\}$

Not every element is a generator: $\{1, 2, 2^2, 2^3, 2^4, 2^5\} = \{1, 2, 4\}$

$\neq \{1, 3, 2 \ldots 5\}$

How do you find a generator?

Order

Z_7^*, $g = 2$. $\{1, 2, 4\}$ order of $2 = 3$. $2^3 = 1 \bmod 7$.

For $g \in (Z_p)^*$ the set $\{1, g, g^2, g^3, \ldots\}$ is called the **group generated by g**, denoted $\langle g \rangle$

g is a generator iff group generated by g is equal to Z_p^*

Def: the **order** of $g \in (Z_p)^*$ is the size of $\langle g \rangle$

$\text{ord}_p(g) = |\langle g \rangle| =$ (smallest $a > 0$ s.t. $g^a = 1$ in Z_p)

Examples: $\text{ord}_7(3) = 6$; $\text{ord}_7(2) = 3$; $\text{ord}_7(1) = 1$

Thm (Lagrange): $\forall g \in (Z_p)^*$: **$\text{ord}_p(g)$** divides $p-1$

Z_p^* $(p-1) = a_1 a_2$ for primes a_1, a_2

$\Rightarrow$ order of any g is 1 or a_1 or a_2 or $a_1 a_2$

To find generator, pick a random element and compute its order. Hit and trial works in practice as long as (p-1) is chosen wisely

* Pick prime p s.t. $p-1 = a_1 a_2$ * Find generator

Euler's generalization of Fermat (1736)

Def: For an integer N define $\varphi(N) = \left|(Z_N)^*\right|$ (Euler's φ func.)

Examples: $\varphi(12) = \left|\{1,5,7,11\}\right| = 4$; $\varphi(p) = p-1$

For $N=p\cdot q$: $\varphi(N) = N-p-q+1 = (p-1)(q-1)$

both primes

Thm (Euler): $\forall\, x \in (Z_N)^*$: $x^{\varphi(N)} = 1$ in Z_N

Example: $5^{\varphi(12)} = 5^4 = 625 = 1$ in Z_{12}

Generalization of Fermat. Basis of the RSA cryptosystem

Hard Problems

Easy problems

- Given composite N and x in Z_N find x^{-1} in Z_N

- Given prime p and polynomial f(x) in $Z_p[x]$

  find x in Z_p s.t. f(x) = 0 in Z_p (if one exists)

  Running time is linear in deg(f) .

* Solve systems of linear equations mod N.

… but many problems are difficult

For any member $a \in \mathbb{Z}_p^*$, $a * a = a^2 \in \mathbb{Z}_p$ is easy.

Intractable problems with primes

e.g. $p = 11$, $g = 2$.

Fix a prime p>2 and g in $(Z_p)^*$ of order q.

Consider the function: $x \mapsto g^x$ in Z_p ($x \to g^x$ is easy.)

(integer)

Now, consider the inverse function: $(g * g * g \ldots)$ x times.

$\text{Dlog}_g(g^x) = x$ where x in {0, …, q-2}

Given y, compute $x \leq q-2$ s.t. $g^x = y$.

Example:

y	1,	2,	3,	4,	5,	6,	7,	8,	9,	10
in $\mathbb{Z}_{11}$:										
$\text{Dlog}_2(\cdot)$:	0	2	?							

Compute x s.t. $2^x = y$

$2^1 = 2$ $2^2 = 4$ $2^3 = 8$ $2^4 = 5$

$2^5 = 10$ $2^6 = 9$ ..... $\sqrt{q}$

Intractable problems with primes

Fix a prime p>2 and g in $(Z_p)^*$ of order q.

Consider the function: $x \longmapsto g^x$ in Z_p

Now, consider the inverse function:

$\mathbf{Dlog_g}(g^x) = x$ **where x in {0, …, q-2}**

Example:

in $\mathbb{Z}_{11}$:	1,	2,	3,	4,	5,	6,	7,	8,	9,	10
$Dlog_2(\cdot)$:	0,	1,	8,	2,	4,	9,	7,	3,	6,	5

DLOG: more generally

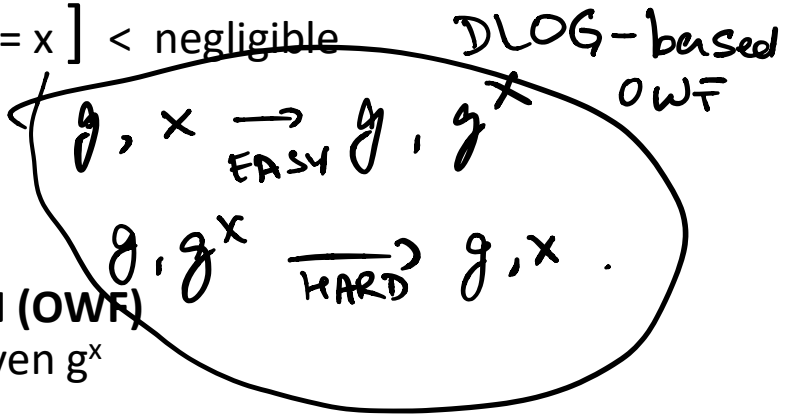
Let $\mathbf{G}$ be a finite cyclic group and $\mathbf{g}$ a generator of G

$$G = \{ 1 , g , g^2 , g^3 , \ \dots \ , g^{q-1} \}$$ (q is called the order of G)

Def: We say that **DLOG is hard in G** if for all efficient alg. A:

$$\Pr_{g \leftarrow G,\ x \leftarrow Z_q} [\ A(G, q, g, g^x) = x\ ] < \text{negligible}$$

Example : $(Z_p)^*$ for large p



This is a candidate **ONE-WAY FUNCTION (OWF)**
Easy to compute g^x but hard to find x given g^x

An application: collision resistance

Choose a group G where Dlog is hard (e.g. $(Z_p)^*$ for large p)

Let q = |G| be a prime. Choose generator g of G and set h = g^s for secret s.

Hash key = (g, h). For $x,y \in \{1,\ldots,q\}$ define $H(x,y) = g^x \cdot h^y$ in G

hard to find z s.t. $h = g^z$.

$H(x,y) = g^x \cdot h^y$ in G

Lemma: finding collision for H(.,.) is as hard as computing $\text{Dlog}_g(h)$

Proof: Suppose we are given a collision $H(x_0,y_0) = H(x_1,y_1)$ (x_0,y_0) & (x_1,y_1) s.t. s.t. $(x_0,y_0) \neq (x_1,y_1)$

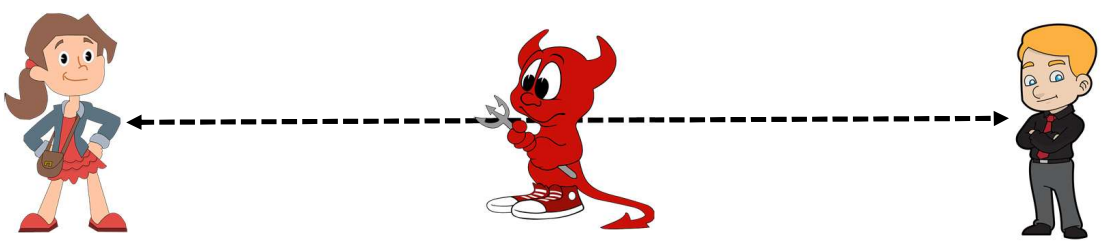
then $g^{x_0} \cdot h^{y_0} = g^{x_1} \cdot h^{y_1} \Rightarrow g^{x_0 - x_1} = h^{y_1 - y_0} \Rightarrow h = g^{\underbrace{(x_0 - x_1)/(y_1 - y_0)}_{z}}$

only works when $y_1 \neq y_0$

s.t. $h = g^z$.

Key Exchange

Setting up a shared key in the presence of an eavesdropper



→ Given (g, g^a, g^b) it should be hard to find g^{ab}.

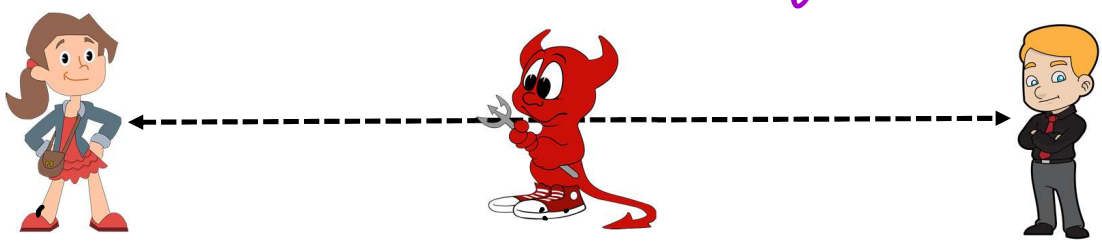
Computational Diffie-Hellman Problem (CDH).

CDH is hard only if DLOG is hard.

Hardness of DLOG ⇏ Hardness of CDH

Space for Discussions - OWF + addnl properties?

Shared secret = g^{ab}



a

$x = g^a$

$x = g^a$

(a, y)

$y = g^b$

$y = g^b$

b

(b, x)

$y^a = (g^b)^a = g^{ab}$

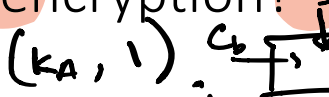
shouldn't be able to compute g^{ab}.

$x^b = (g^a)^b = g^{ab}$.

private key encryption + rerandomization ⇒ public key encryption

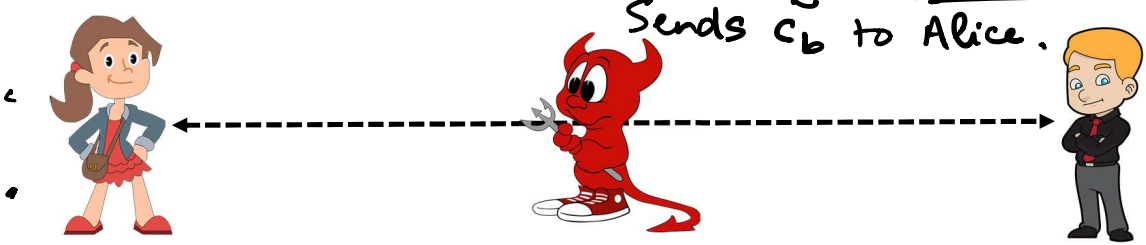
Space for Discussions – rerandomizable encryption?

Soln 2. Alice publishes $c_0 = E(k_A, 0)$, $c_1 = E(k_A, 1)$.



Bob wants to send b. Picks c_b, rerandomizes → $\tilde{c}_b$.

Sends $\tilde{c}_b$ to Alice.



Soln 1.

$c_A = E(k_A, x)$

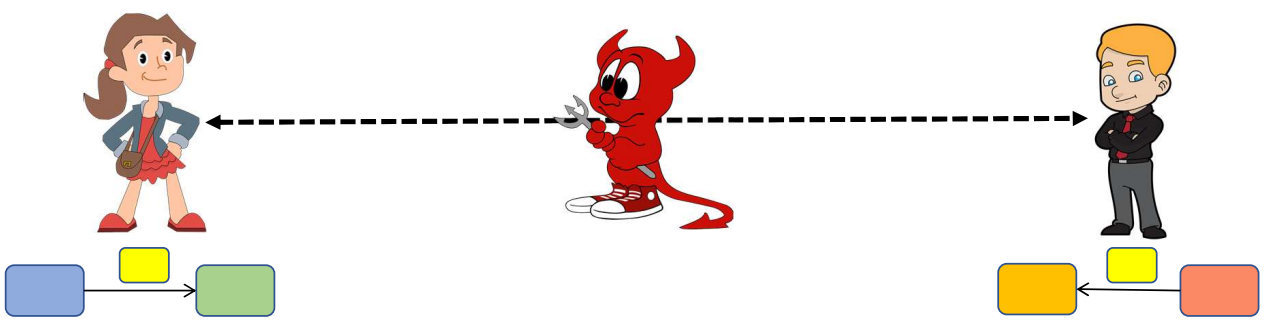
c_A → (⊕ Bob secret key) → $c_{AB} = E(k_B(E(k_A, x)))$

c_{AB} ←

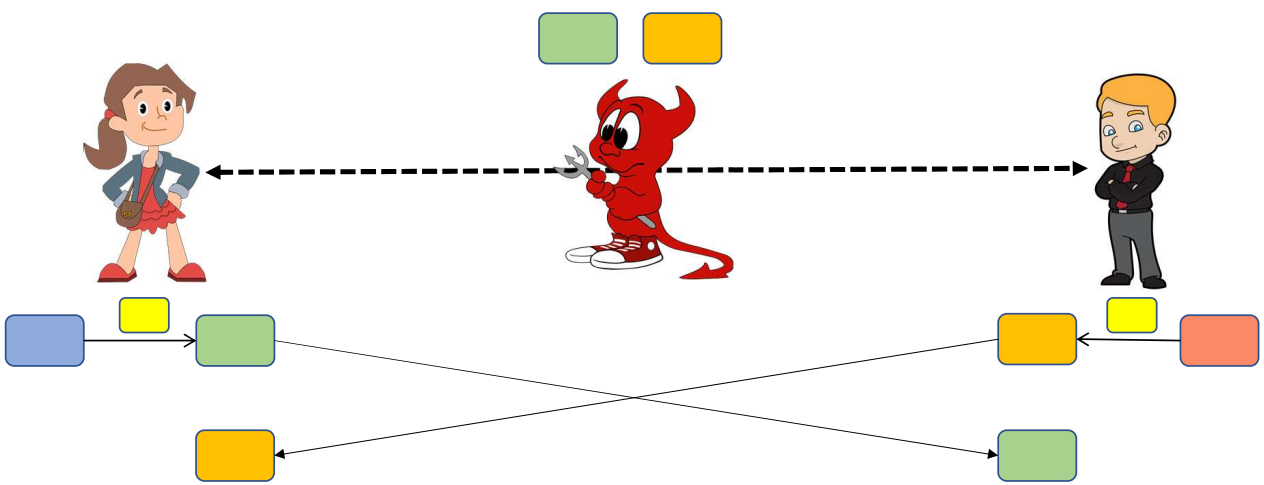
$D(k_A(E(k_B(E(k_A, x)))))$ $\xrightarrow{c_B}$ $D(k_B(D(k_A(E(k_B(E(k_A^{-1}))))))$

$c_B = x \oplus$ Bob secret key.

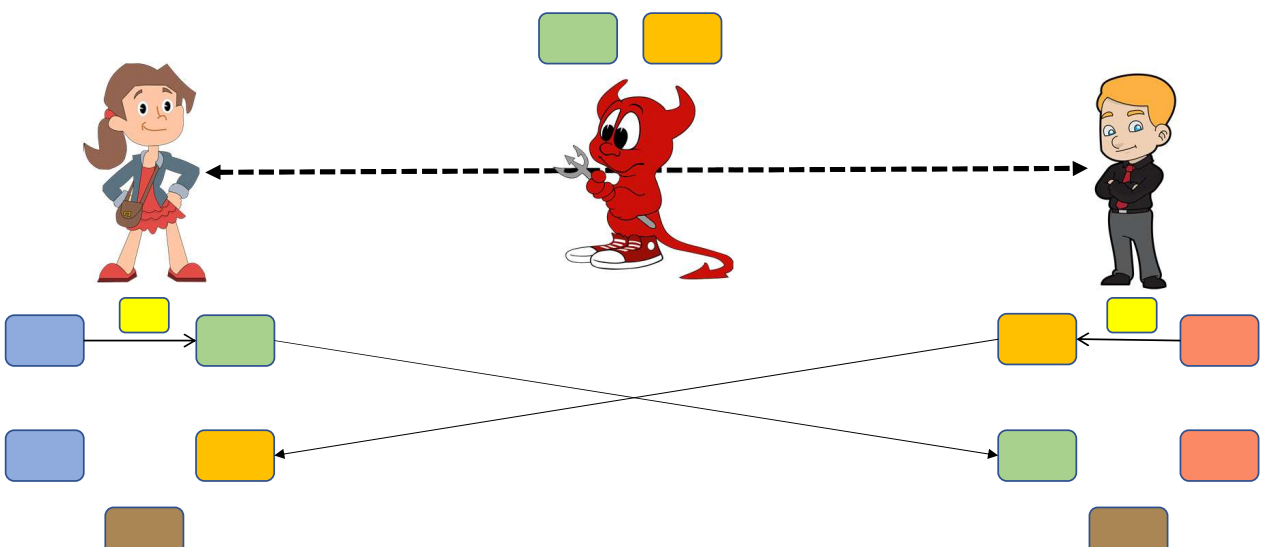
Setting up a shared key in the presence of an eavesdropper



Setting up a shared key in the presence of an eavesdropper



Setting up a shared key in the presence of an eavesdropper



The Diffie-Hellman protocol (informally)

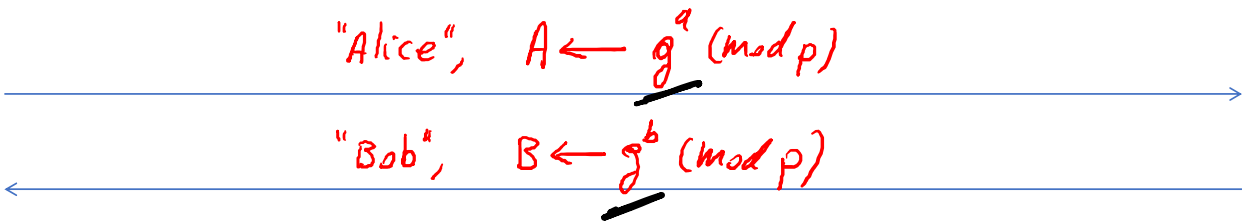
CDH

Fix a large prime p (e.g. 600 digits)

Fix an integer g in {1, …, p-1}

Alice **Bob**

choose random **a** in {1,…,p-1} choose random **b** in {1,…,p-1}



$B^a \pmod p = (g^b)^a = k_{AB} = g^{ab} \pmod p = (g^a)^b = A^b \pmod p$

Security (much more on this later)

Eavesdropper sees: p, g, $A=g^a \pmod p$, and $B=g^b \pmod p$

Can she compute $g^{ab} \pmod p$??

More generally: define $DH_g(g^a, g^b) = g^{ab} \pmod p$

How hard is the DH function mod p?

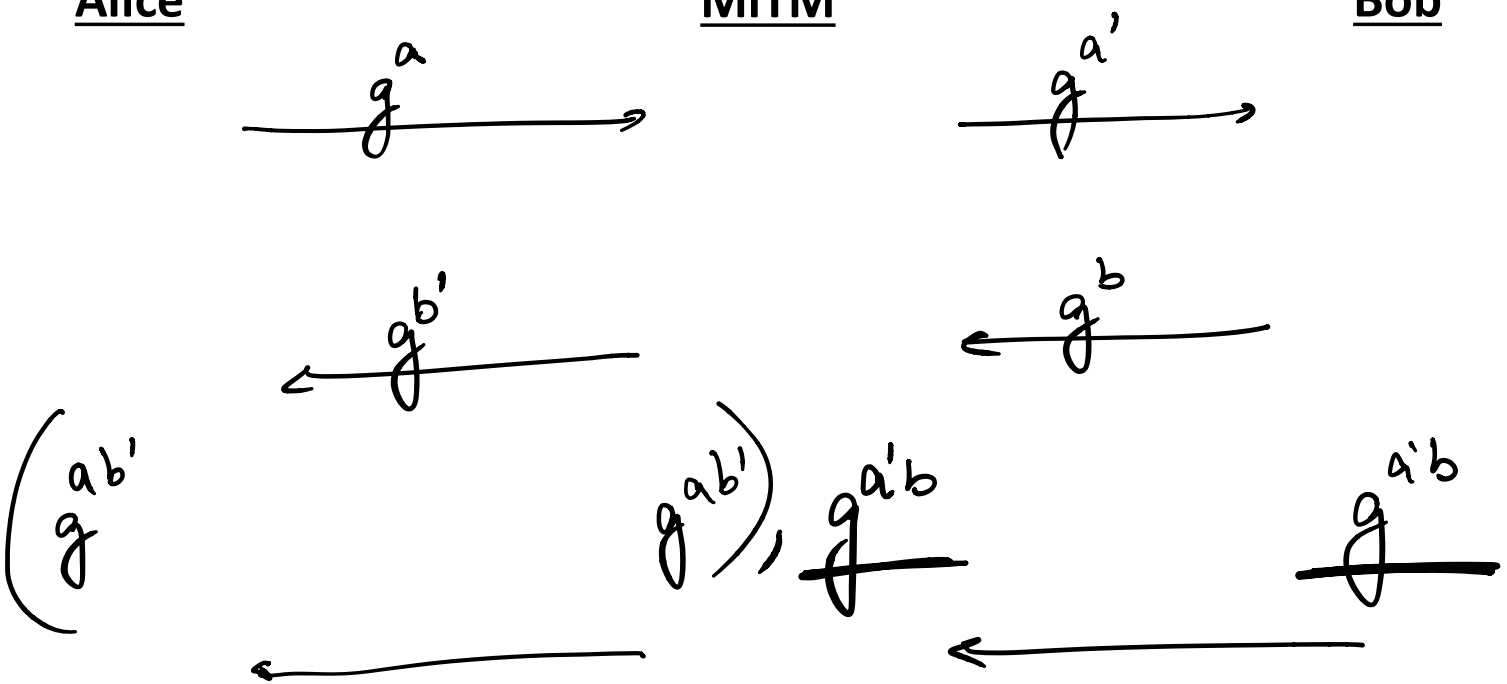
If DH is hard then DLOG is hard. If DLOG is hard then DH may or may not be hard. Both believed to be hard in Z_p^*.

Elliptic curve Diffie-Hellman

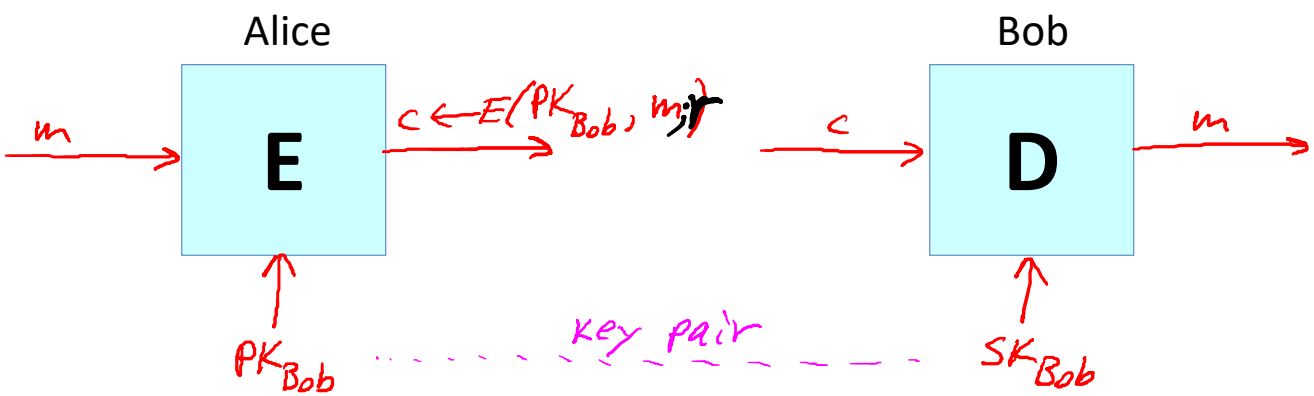
$\mathbb{Z}_p^*$ is replaced by a different group.

Insecure against man-in-the-middle

As described, the protocol is insecure against **active** attacks



Public key encryption



PK: public key, SK: secret key

Public key encryption

Def: a public-key encryption system is a triple of algs. (G, E, D)

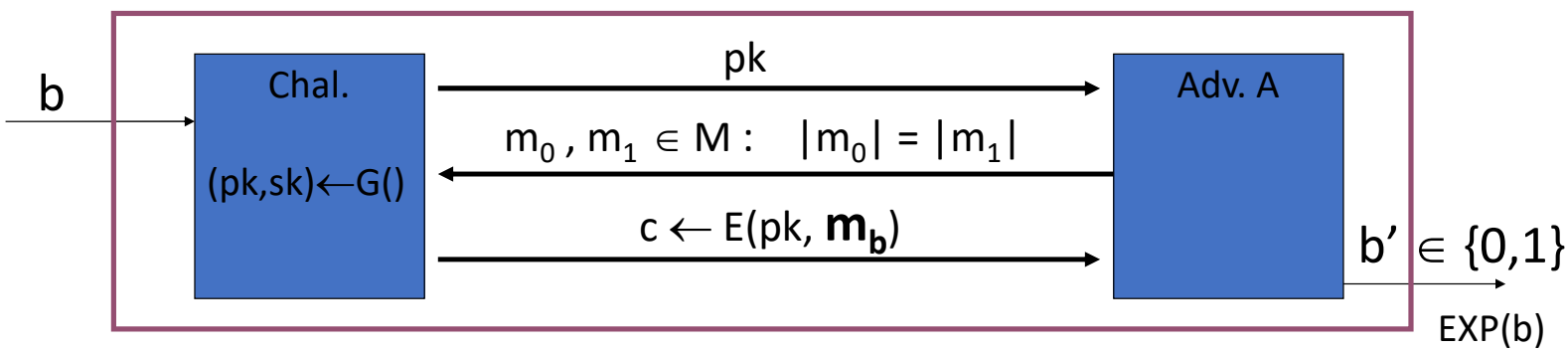
- G(): randomized alg. outputs a key pair (pk, sk)
- E(pk, m): randomized alg. that takes $m \in M$ and outputs $c \in C$
- D(sk,c): det. alg. that takes $c \in C$ and outputs $m \in M$ or $\perp$

Consistency: $\forall$(pk, sk) output by G :

$$\forall m \in M: \quad D(sk, E(pk, m)) = m$$

Semantic Security

For b=0,1 define experiments EXP(0) and EXP(1) as:



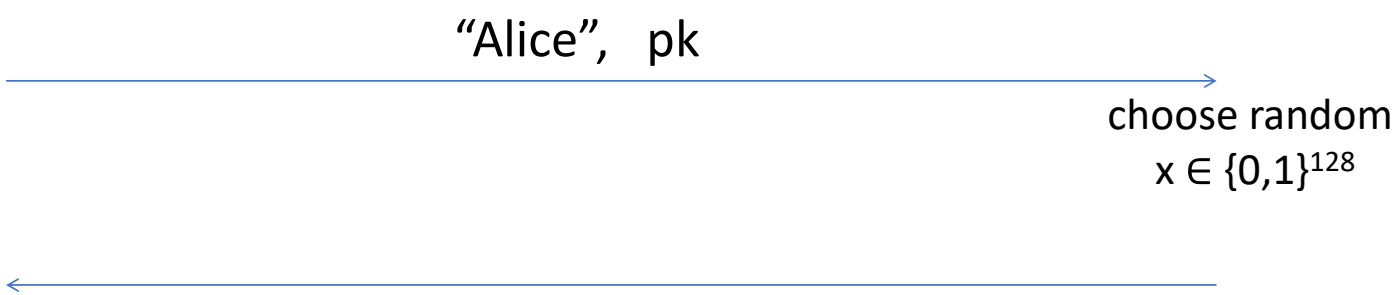
Def: $\mathbb{E}$ =(G,E,D) is sem. secure (a.k.a IND-CPA) if for all efficient A:

$$Adv_{SS}[A,\mathbb{E}] = \left| Pr[EXP(0)=1] - Pr[EXP(1)=1] \right| < \text{negligible}$$

Establishing a shared secret

Alice **Bob**

(pk, sk) ⟵ G()



Security (eavesdropping)

Adversary sees **pk, E(pk, x)** and wants **x** $\in M$

Semantic security $\Rightarrow$

adversary cannot distinguish

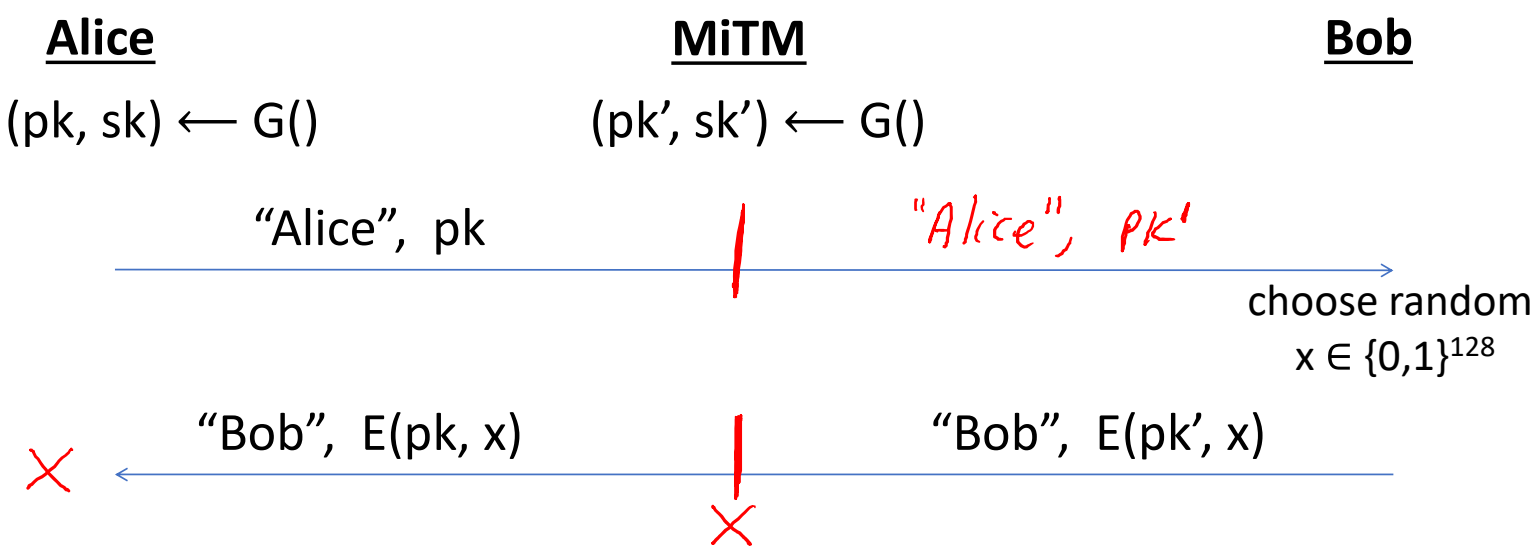
$\{ pk, E(pk, x), x \}$ from $\{ pk, E(pk, x), rand \in M \}$

$\Rightarrow$ can derive session key from x.

Note: protocol is vulnerable to man-in-the-middle

Insecure against man in the middle

As described, the protocol is insecure against **active** attacks



Intractable problems with composites

Consider the set of integers: (e.g. for n=1024)

$$\mathbb{Z}_{(2)}(n) \;:=\; \{\, N = p \cdot q \;\text{ where }\; p,q \;\text{ are n-bit primes} \,\}$$

Problem 1: Factor a random N in $\mathbb{Z}_{(2)}(n)$ (e.g. for n=1024)

Problem 2: Given a polynomial **f(x)** where degree(f) > 1

and a random N in $\mathbb{Z}_{(2)}(n)$

find x in $\mathbb{Z}_N$ s.t. f(x) = 0 in $\mathbb{Z}_N$

The factoring problem

Gauss (1805): *"The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic."*

Best known alg. (NFS): run time exp($\tilde{O}(\sqrt[3]{n})$) for n-bit integer

Current world record: **RSA-768** (232 digits)

- Work: two years on hundreds of machines
- Factoring a 1024-bit integer: about 1000 times harder

$\Rightarrow$ likely possible this decade

Summary

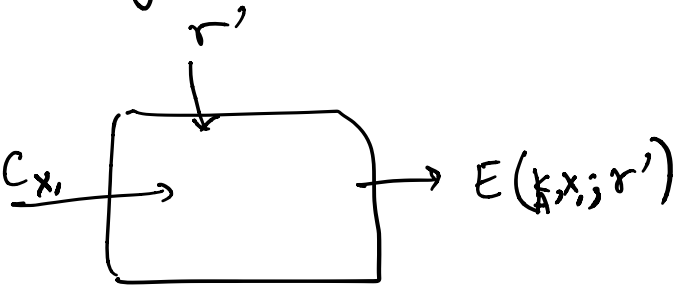
- Key concepts in number theory
- Hardness of discrete logarithm, factoring
- Diffie-Hellman key exchange from hardness of DDH
- Public key encryption => shared key derivation (called key exchange)

Rerandomizable Encryption

$x = x_1 \ldots x_n$

$\begin{cases} c_0 = E(k_A, 0; r_0) \text{ is encryption of } 0 \\ c_1 = E(k_A, 1; r_1) \text{ is encryption of } 1. \end{cases}$

Rerandomization:



Decryption: $D(k_A, E(k_A, x_i; r')) = x_i$

Homomorphic Encryption: $\underbrace{d_0}_{=E(k, m_0; r_0)} \quad \underbrace{d_1}_{=E(k, m_1; r_1)}$

$d = E(k, m_0 \oplus m_1; r_0 \oplus r_1)$