CIS 5560

Cryptography
Lecture 14

Course website:
pratyushmishra.com/classes/cis-5560-s25/

Slides adapted from Dan Boneh and Vinod Vaikuntanathan

Recap of Last Lecture(s)

- Number Theory refresher
  - Arithmetic modulo primes
  - Fermat's Little Theorem
  - Cyclic groups
  - Discrete Logarithms
- Key Exchange
  - Merkle puzzles
  - Diffie—Hellman
    - Computational Diffie—Hellman Problem

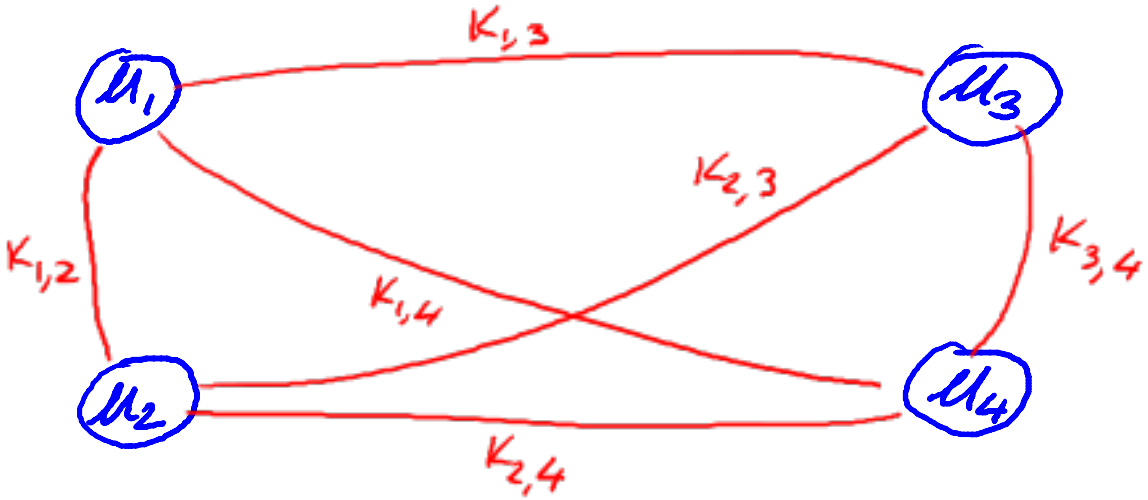
The Multiplicative Group $\mathbb{Z}_p^*$

$\mathbb{Z}_p^*$: $(\{1,\ldots,p-1\}$, group operation: $\bullet$ mod $p)$

- Computing the group operation is easy.
- Computing inverses is easy: Extended Euclid.
- Exponentiation (given $g \in \mathbb{Z}_p^*$ and $x \in \mathbb{Z}_{p-1}$, find g^x mod p) is easy: **Repeated Squaring Algorithm.**
- The discrete logarithm problem (given a generator g and $h \in \mathbb{Z}_p^*$, find $x \in \mathbb{Z}_{p-1}$ s.t. $h = g^x$ mod p) is **hard**, to the best of our knowledge!

Key management

Problem: n users. Storing mutual secret keys is difficult



Total: O(n) keys per user

Key question

Can we generate shared keys without an **online** trusted 3rd party?

Answer: yes!

Starting point of public-key cryptography:

- Merkle (1974), Diffie-Hellman (1976), RSA (1977)

- More recently: ID-based enc. (BF 2001), Functional enc. (BSW 2011)

Merkle Puzzles (1974)

Answer: yes, but very inefficient

Main tool: puzzles

- Problems that can be solved with some effort
- Example: E(k,m) a symmetric cipher with $k \in \{0,1\}^{128}$
  - **puzzle(P) = E(P, "message")** where $P = 0^{96} \,||\, b_1 \ldots b_{32}$
  - Goal: find P by trying all 2^{32} possibilities

Merkle puzzles

Alice: prepare 2^{32} puzzles

- For i=1, …, 2^{32} choose random $\mathbf{P_i} \in \{0,1\}^{32}$ and $\mathbf{x_i, k_i} \in \{0,1\}^{128}$

  set $\text{puzzle}_i \leftarrow E(0^{96} \,||\, \mathbf{P_i}, \text{“Puzzle \# } \mathbf{x_i}\text{”} \,||\, \mathbf{k_i})$

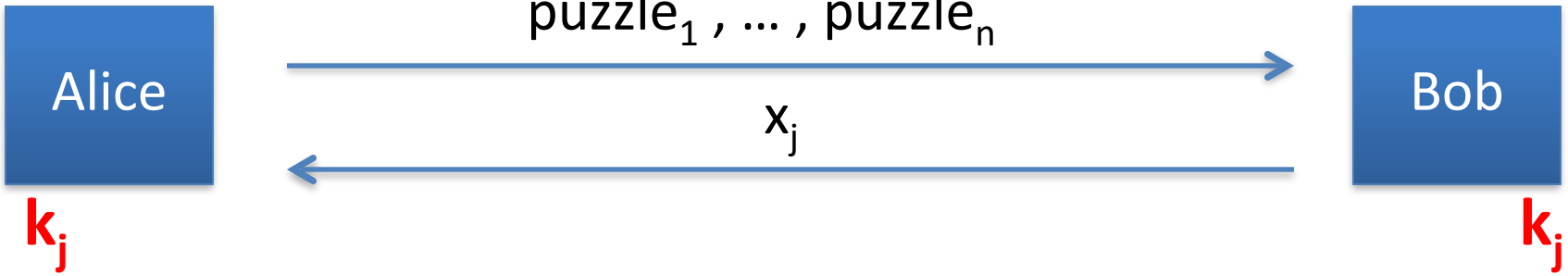
- Send $\text{puzzle}_1, \ldots, \text{puzzle}_{2^{32}}$ to Bob

Bob: choose a random puzzle_j and solve it. Obtain (x_j, k_j).

- Send x_j to Alice

Alice: lookup puzzle with number x_j. Use k_j as shared secret

In a figure



Alice's work: O(n) (prepare n puzzles)

Bob's work: O(n) (solve one puzzle)

Eavesdropper's work: $O(n^2)$ (e.g. 2^{64} time)

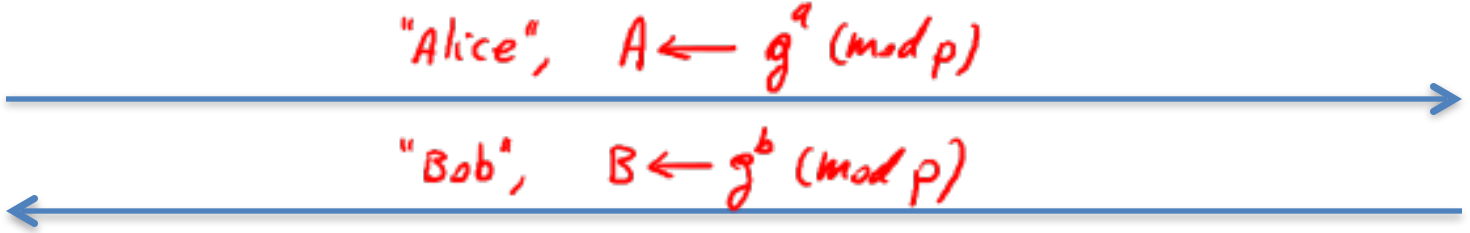
The Diffie-Hellman protocol (informally)

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

Fix generator g of $\mathbb{Z}_p^*$

Alice **Bob**

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

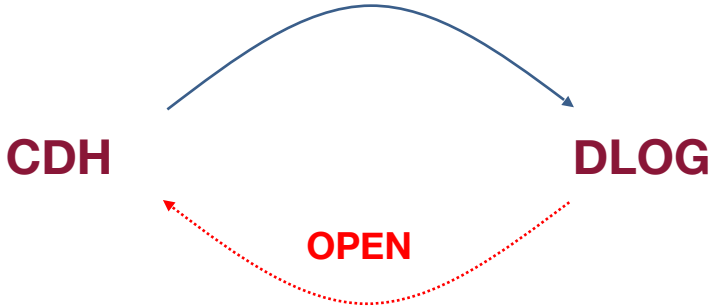


$B^a \pmod p = (g^b)^a = k_{AB} = g^{ab} \pmod p = (g^a)^b = A^b \pmod p$

Computational Diffie-Hellman (CDH) Assumption

W.r.t. a random prime: for every p.p.t. algorithm A, there is a negligible function μ s.t.

$$\Pr\left[\begin{array}{l} p \leftarrow PRIMES_n; g \leftarrow GEN\left(\mathbb{Z}_p^*\right); \\ x, y \leftarrow \mathbb{Z}_{p-1}: A\left(p, g, g^x, g^y\right) = g^{xy} \end{array}\right] = \mu(n)$$

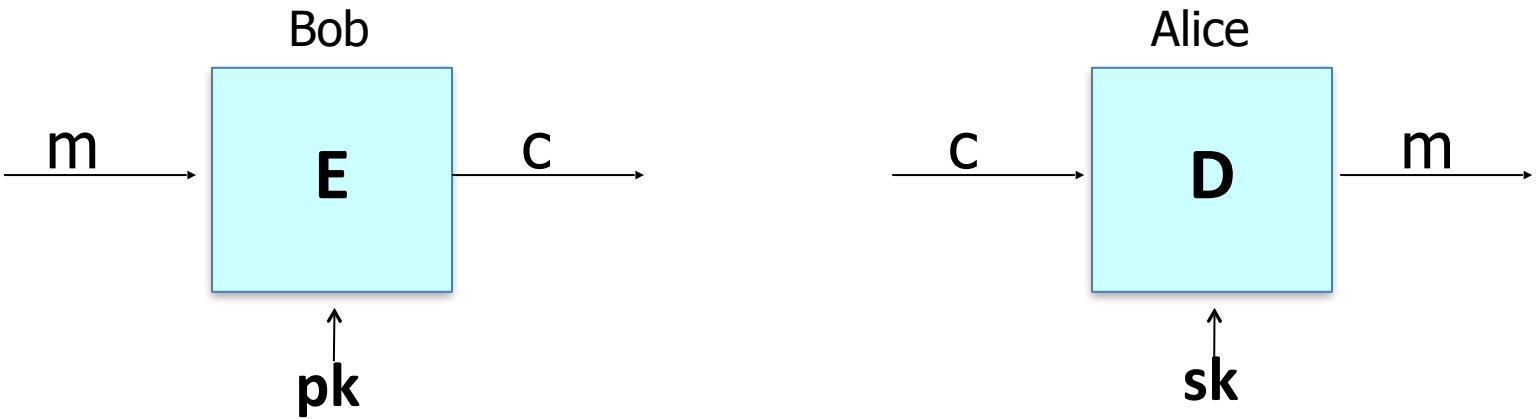


Today's Lecture

- Public Key Encryption
  - El Gamal Encryption
    - Computational Diffie—Hellman Problem
  - RSA Encryption
    - Arithmetic modulo composites
    - Factoring

Public key encryption

Alice: generates (PK, SK) and gives PK to Bob



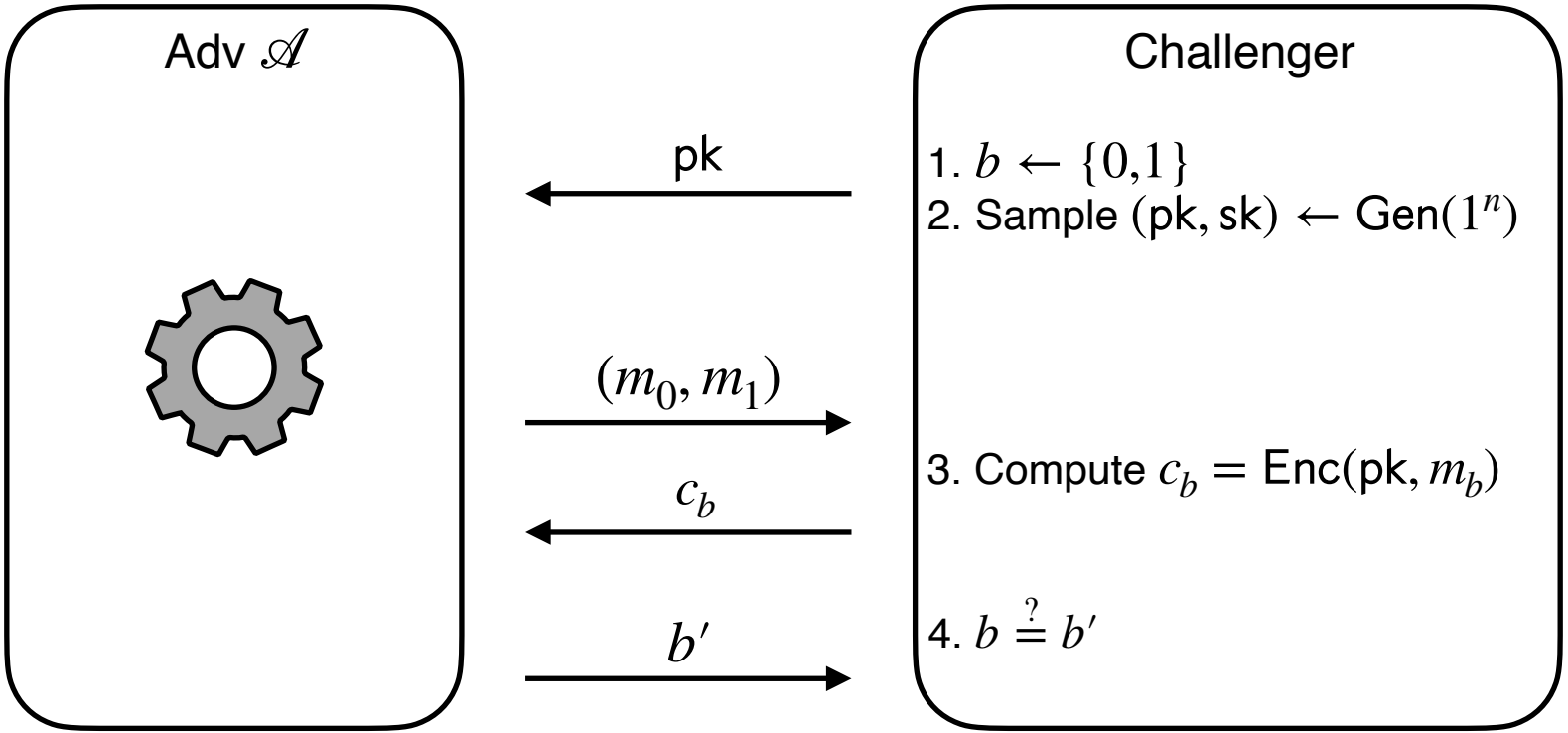
Public key encryption

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

- Gen(): randomized alg. outputs a key pair $(\mathsf{pk}, \mathsf{sk})$
- Enc(pk, m): randomized alg. that takes $m \in \mathcal{M}$ and outputs $c \in \mathcal{C}$
- Dec(sk, c): deterministic alg. that takes $c \in \mathcal{C}$ and outputs $m \in \mathcal{M} \cup \{ \perp \}$

Correctness: $\forall(\mathsf{pk}, \mathsf{sk})$ output by $\mathsf{Gen}(), \forall m \in \mathcal{M}, \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m$

Security: IND-CPA for PKE



$$\Pr[b = b'] = 1/2 + \text{negl}(n)$$

Security: IND-CPA for PKE

For all PPT adversaries $\mathcal{A}$, the following holds:

$$\Pr\left[b = \mathcal{A}(\mathsf{Enc}(\mathsf{pk}, m_b)) \;\middle|\; \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n) \\ \text{Sample } b \leftarrow \{0,1\} \\ (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk}) \end{array} \right] \leq \mathsf{negl}(n)$$

How does it relate to symmetric-key IND-CPA?

Recall: for symmetric ciphers we had two security notions:

- One-time security and many-time security (CPA)
- We showed that one-time security does not imply many-time security

For public key encryption:

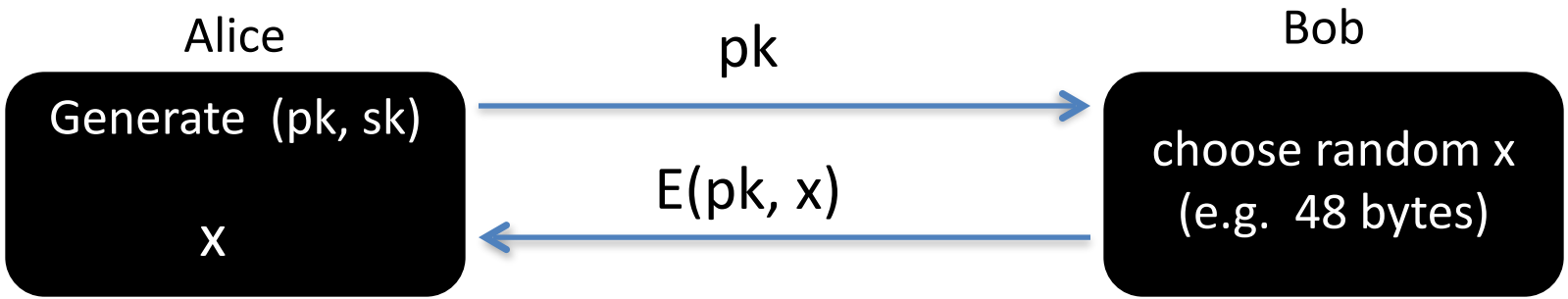
- One-time security ⇒ many-time security (CPA)

  (follows from the fact that attacker can encrypt by themselves)

- Public key encryption **must** be randomized
  - Q: why not stateful?

Applications

Session setup (for now, only eavesdropping security)

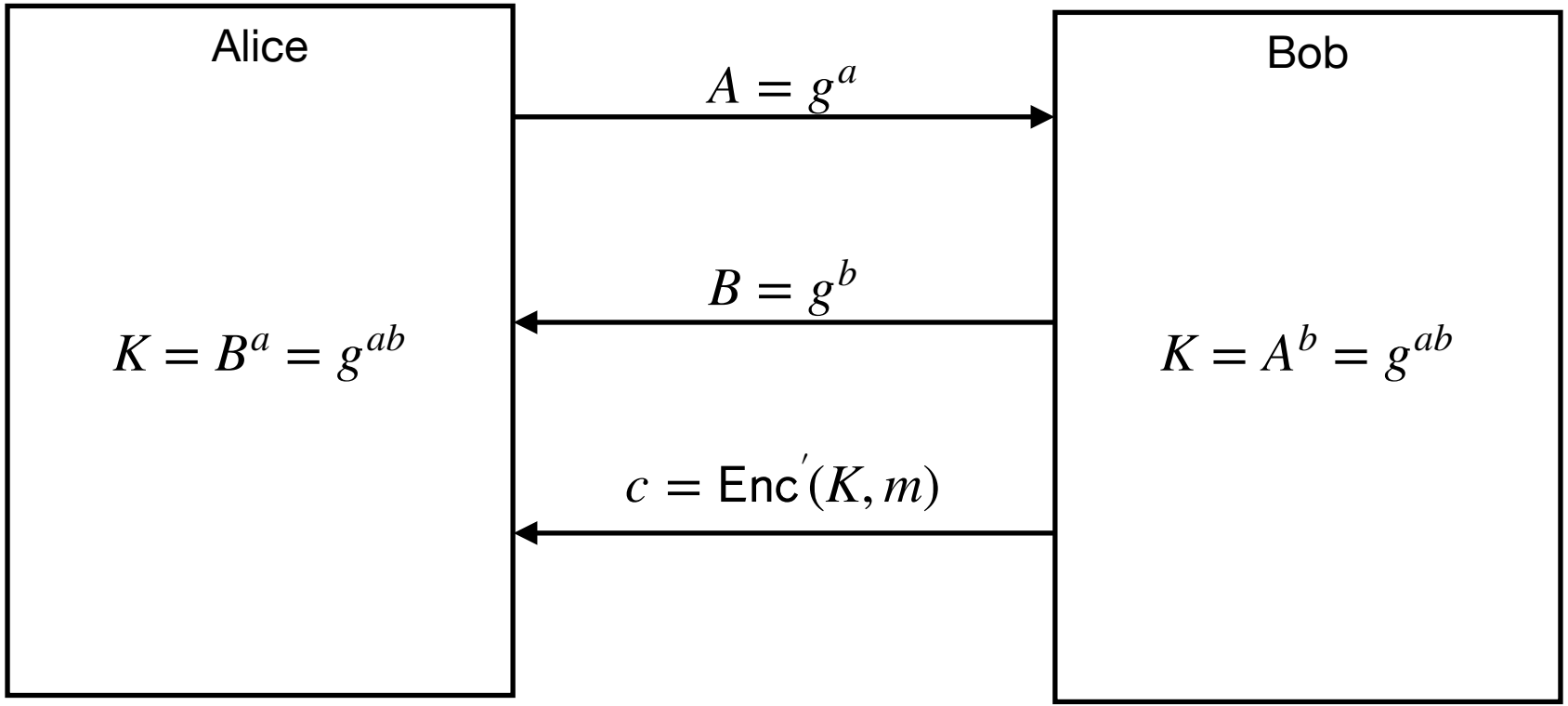


Non-interactive applications: (e.g. Email)

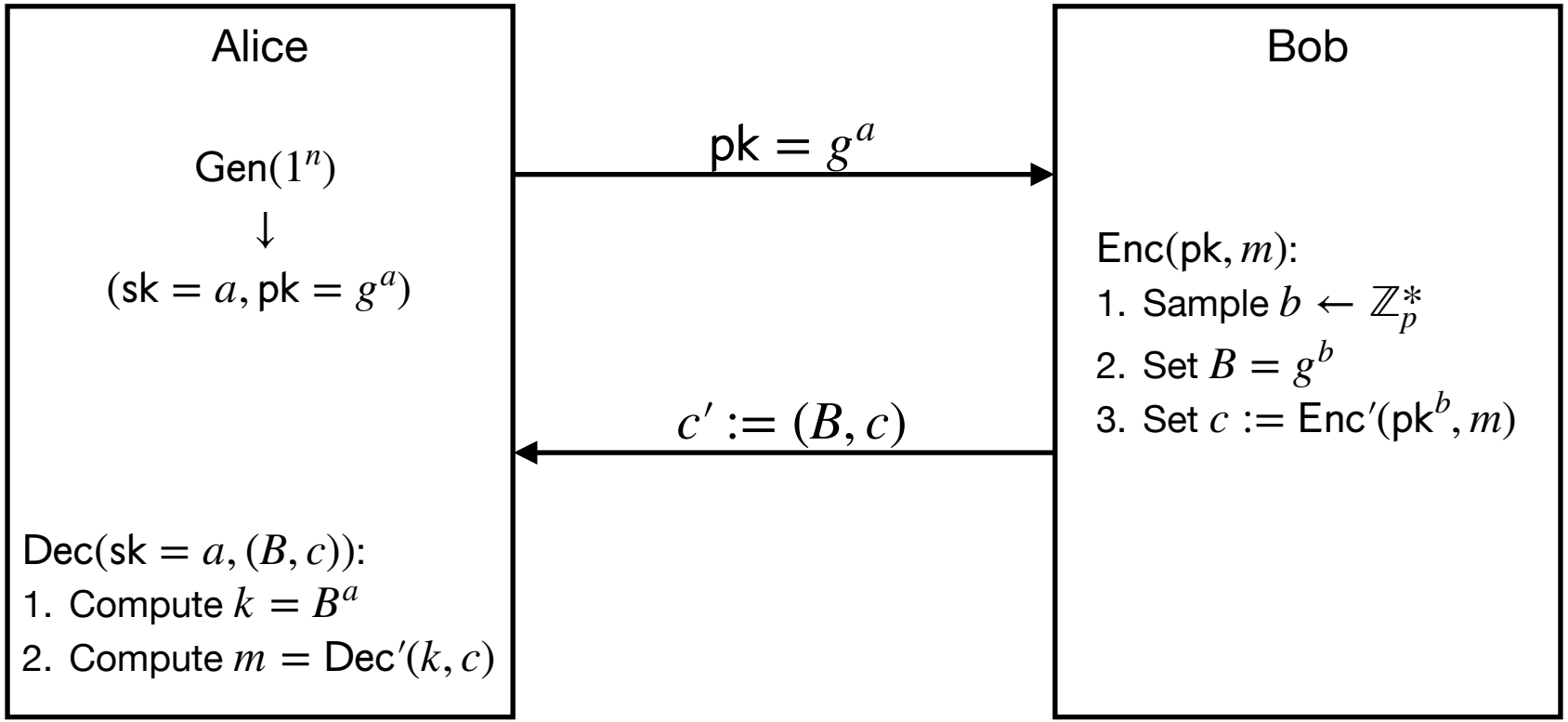
- Bob sends email to Alice encrypted using pk_{alice}
- Note: Bob needs pk_{alice} (public key management)

Constructions of PKE

Recall: DH Key Exchange



Convert DH → PKE



The Elgamal system (an abstract view)

- $\mathbb{G}$: finite cyclic group of prime order p with generator g
- $(\mathsf{Enc}', \mathsf{Dec}')$: symmetric-key encryption with keyspace $\mathscr{K} = \mathbb{G}$

Gen(1^n):
1. Sample $a \leftarrow \mathbb{Z}_p^*$
2. Output $(\mathsf{sk} = a, \mathsf{pk} = g^a)$

Enc(pk, m):
1. Sample $b \leftarrow \mathbb{Z}_p^*$
2. Set $B = g^b$
3. Set $c := \mathsf{Enc}'(\mathsf{pk}^b, m)$
4. Output $c' = (B, c)$

Dec(sk $= a$, (B, c)):
1. Compute $k = B^a$
2. Output $m = \mathsf{Dec}'(k, c)$

What choice of $(\mathsf{Enc}', \mathsf{Dec}')$?

How to prove security?

Q1: Choice of $(\mathsf{Enc}', \mathsf{Dec}')$: OTP?

- $\mathbb{G}$: finite cyclic group of prime order p with generator g
- Key idea: One-Time Pad works not just with $\{0,1\}^n$ and XOR, but with *any group*
  - $\mathsf{Gen}'(1^n)$: Sample $r \leftarrow \mathbb{Z}_p$, and output g^r
  - $\mathsf{Enc}'(k = g^r, m \in \mathbb{G})$: Output $c = k \cdot m \in \mathbb{G}$
  - $\mathsf{Dec}'(k = g^r, c \in \mathbb{G})$: Output $m = k^{-1} \cdot c \in \mathbb{G}$

Correctness: $\mathsf{Dec}'(k, \mathsf{Enc}'(k, m)) = k \cdot m \cdot k^{-1} = m$

Security: Goal: $\forall m, m' \in \mathbb{G}, c \in \mathbb{G}, \Pr_{k \leftarrow \mathbb{G}}[\mathsf{Enc}(k, m) = c] = \Pr_{k \leftarrow \mathbb{G}}[\mathsf{Enc}(k, m') = c]$

Exercise: prove this (try to adapt proof from Lecture 1)

The Elgamal system (a concrete view)

- $\mathbb{G}$: finite cyclic group of prime order p with generator g
- $(\mathsf{Enc}', \mathsf{Dec}')$: symmetric-key encryption with keyspace $\mathscr{K} = \mathbb{G}$

Gen(1^n):

1. Sample $a \leftarrow \mathbb{Z}_p^*$
2. Output $(\mathsf{sk} = a, \mathsf{pk} = g^a)$

Enc(pk, m):

1. Sample $b \leftarrow \mathbb{Z}_p^*$
2. Set $B = g^b$
3. Set $c := \mathsf{Enc}'(\mathsf{pk}^b, m)$
4. Output $c' = (B, c)$

Dec(sk $= a$, (B, c)):

1. Compute $k = B^a$
2. Output $m = \mathsf{Dec}'(k, c)$

What choice of $(\mathsf{Enc}', \mathsf{Dec}')$?

How to prove security?

The Elgamal system (a concrete view)

- $\mathbb{G}$: finite cyclic group of prime order p with generator g
- $(\mathsf{Enc}', \mathsf{Dec}')$: symmetric-key encryption with keyspace $\mathscr{K} = \mathbb{G}$

Gen(1^n):	Enc(pk, m):	Dec(sk $= a$, (B, c)):
1. Sample $a \leftarrow \mathbb{Z}_p^*$	1. Sample $b \leftarrow \mathbb{Z}_p^*$	1. Compute $k = B^a$
2. Output (sk $= a$, pk $= g^a$)	2. Set $B = g^b$	2. Output $m = k^{-1}c$
	3. Set $c := m \cdot \mathsf{pk}^b = mg^{ab}$	$= cg^{-ab}$
	4. Output $c' = (B, c)$	$= mg^{ab}g^{-ab}$

What choice of $(\mathsf{Enc}', \mathsf{Dec}')$?

How to prove security?

Problem:
OTP uses random group element

But we only have g^{ab}!

Is this a problem? Isn't g^{ab} also random?

Problem: adversary *also* sees g^a and g^b!

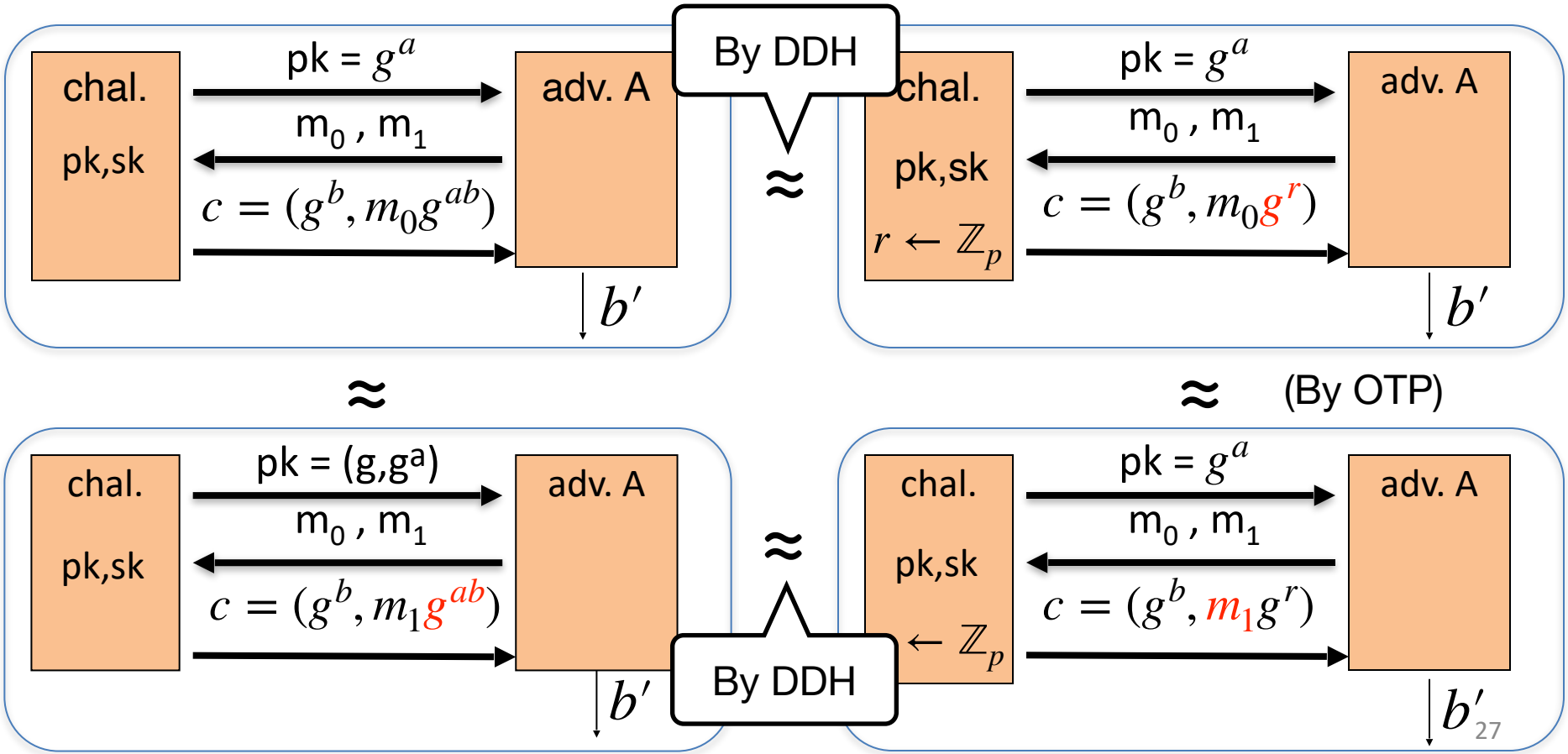
New assumption: Decisional Diffie—Hellman

Roughly, (g^a, g^b, g^{ab}) is indistinguishable from (g^a, g^b, g^r)

Formally, the following two distributions are computationally indistinguishable:

$\{(g^a, g^b, g^{ab})\}_{a,b \leftarrow \mathbb{Z}_p}$ and $\{(g^a, g^b, g^r)\}_{a,b,r \leftarrow \mathbb{Z}_p}$

Elgamal is semantically secure under DDH



The Elgamal system (a modern view)

- $\mathbb{G}$: finite cyclic group of prime order p with generator g
- $(\mathsf{Enc}', \mathsf{Dec}')$: what about arbitrary keyspace $\mathcal{K}$?
- New ingredient: "Random"-ish hash function $H : \mathbb{G} \to \mathcal{K}$

Gen(1^n):
1. Sample $a \leftarrow \mathbb{Z}_p^*$
2. Output $(\mathsf{sk} = a, \mathsf{pk} = g^a)$

Enc(pk, m):
1. Sample $b \leftarrow \mathbb{Z}_p^*$
2. Set $k := H(g^{ab})$
3. Set $c \leftarrow \mathsf{Enc}(k, m)$
4. Output $c' = (g^b, c)$

Dec(sk $= a, (B, c)$):
1. Compute $k = H(B^a)$
2. Output $m = \mathsf{Dec}'(k, c)$

New assumption: Hash-DDH

Roughly, $(g^a, g^b, H(g^{ab}))$ is indistinguishable from (g^a, g^b, R)

Formally, the following two distributions are computationally indistinguishable:

$\{(g^a, g^b, H(g^{ab}))\}_{a,b \leftarrow \mathbb{Z}_p}$ and $\{(g^a, g^b, R)\}_{a,b \leftarrow \mathbb{Z}_p, R \leftarrow \mathcal{K}}$

Q: If DDH is hard, is H-DDH hard?

Q: If H-DDH is hard, is DDH hard?

Suppose $K = \{0,1\}^{128}$ and

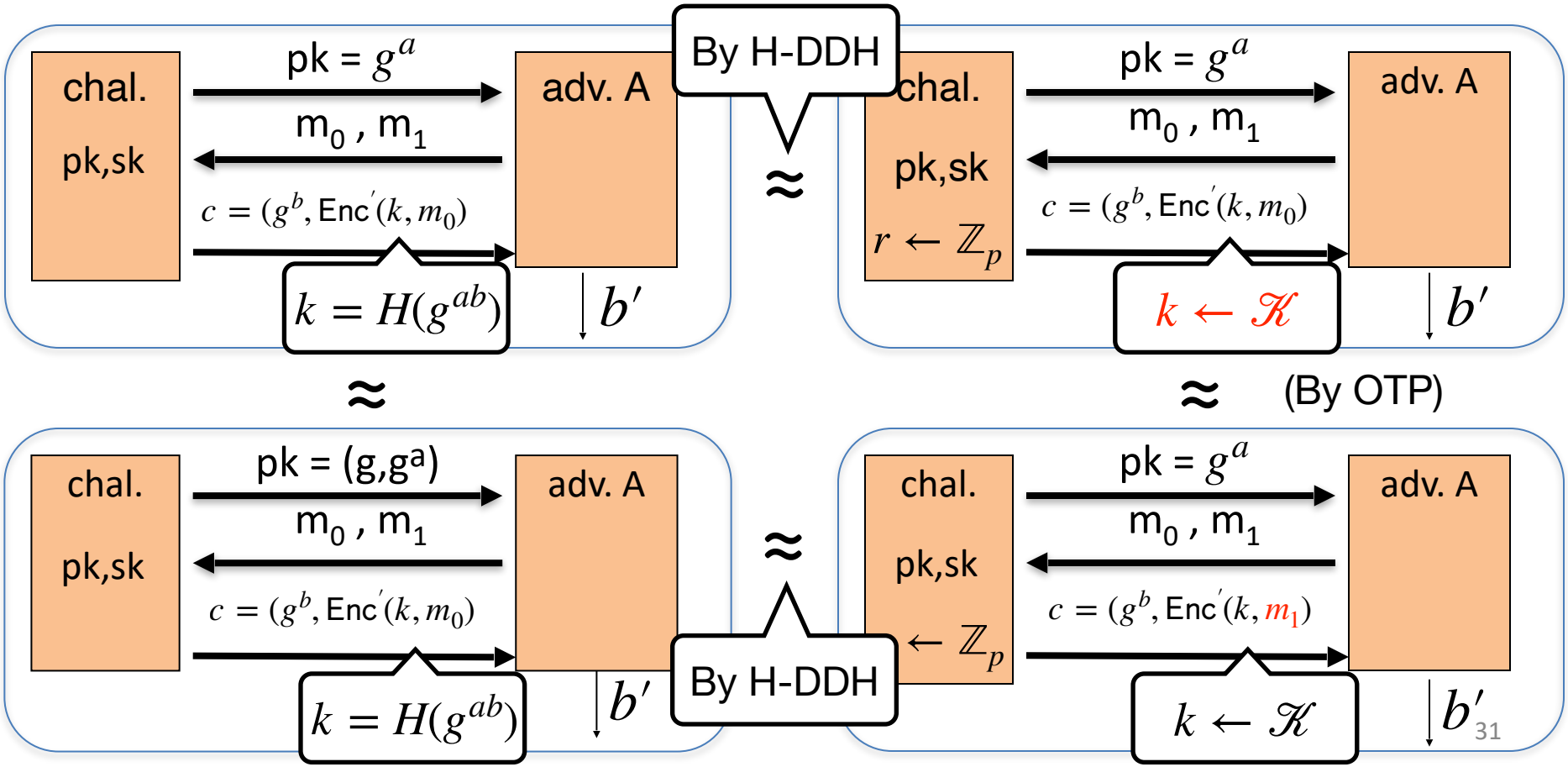
H: G $\longrightarrow$ K only outputs strings in K that begin with 0

(i.e. for all y: msb(H(y))=0)

Can Hash-DH hold for (G, H) ?

- ○ Yes, for some groups G
- ⇒ ○ No, Hash-DH is easy to break in this case
- ○ Yes, Hash-DH is always true for such H

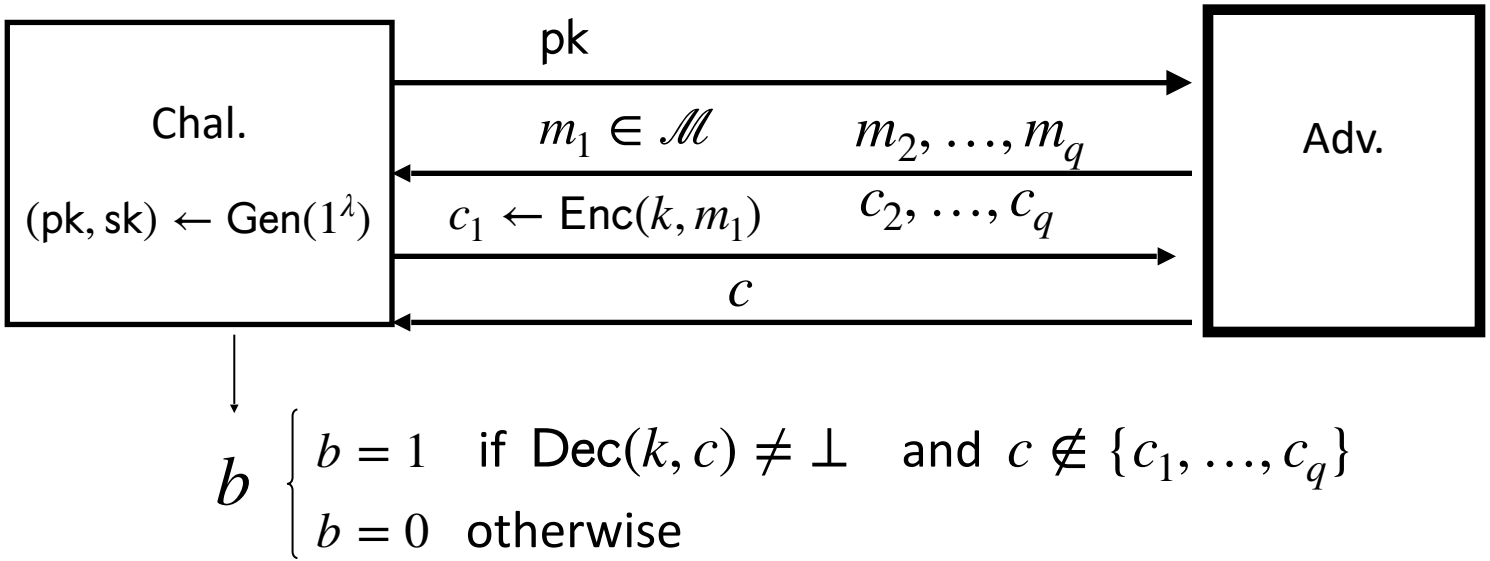
Elgamal is semantically secure under H-DDH



What about active attacks?

What about security against active attacks?

Can we achieve ciphertext integrity?



Def: (Gen, Enc, Dec) has **ciphertext integrity** if for all PPT A:

$$\mathsf{Adv}_{\mathrm{CI}}[A] = \Pr[b = 1] = \mathsf{negl}(\lambda)$$

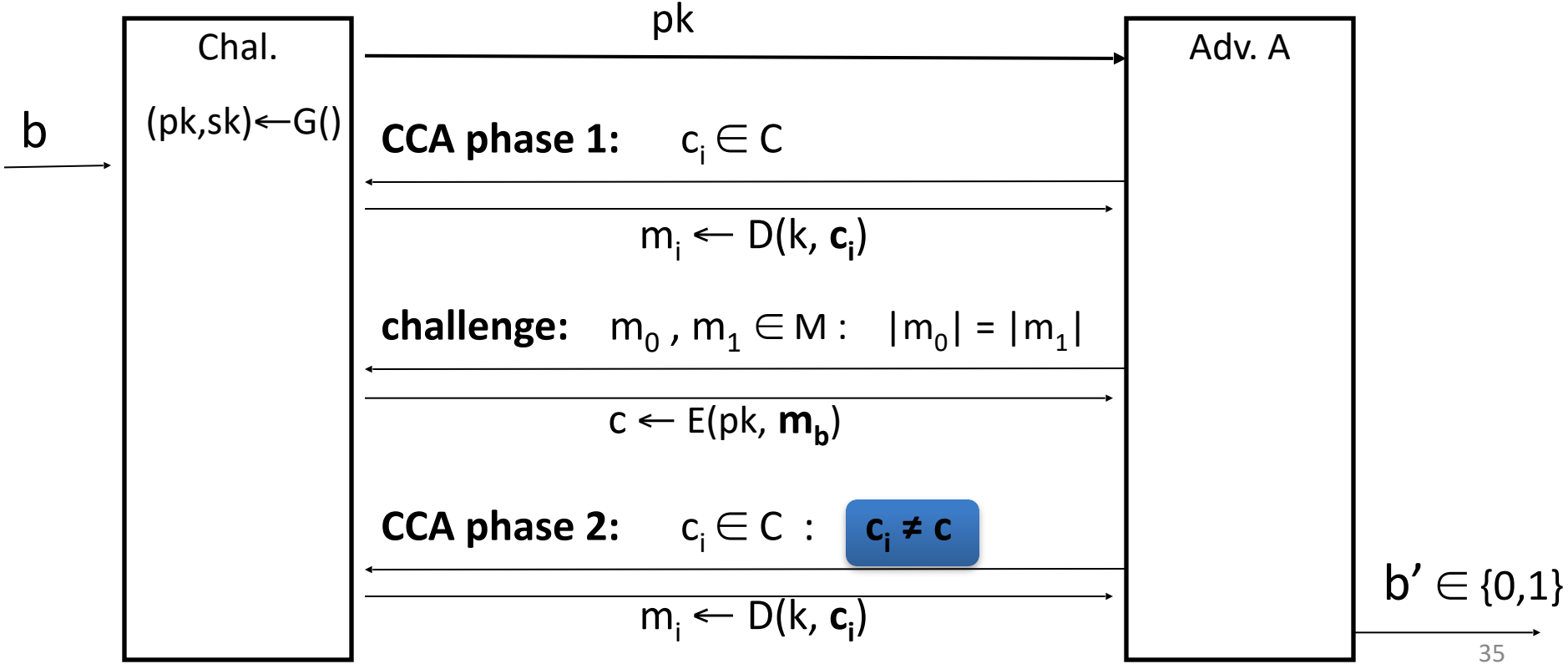
Problem

In public-key settings:

- Attacker **can** *always* create new ciphertexts using pk !!
- So instead: we directly require chosen ciphertext security

(pub-key) Chosen Ciphertext Security: definition

E = (G,E,D) public-key enc. over (M,C). For b=0,1 define EXP(b):



Chosen ciphertext security: definition

Def: E is CCA secure (a.k.a IND-CCA) if for all efficient A:

$Adv_{CCA}[A,E] = |Pr[EXP(0)=1] - Pr[EXP(1)=1]|$ is negligible.

b

ElGamal chosen ciphertext security?

Security Theorem:

If **IDH** holds in the group G, (E_s, D_s) provides auth. enc.

and **H:** $G^2 \longrightarrow K$ is a "random oracle"

then **ElGamal** is CCA^{ro} secure.

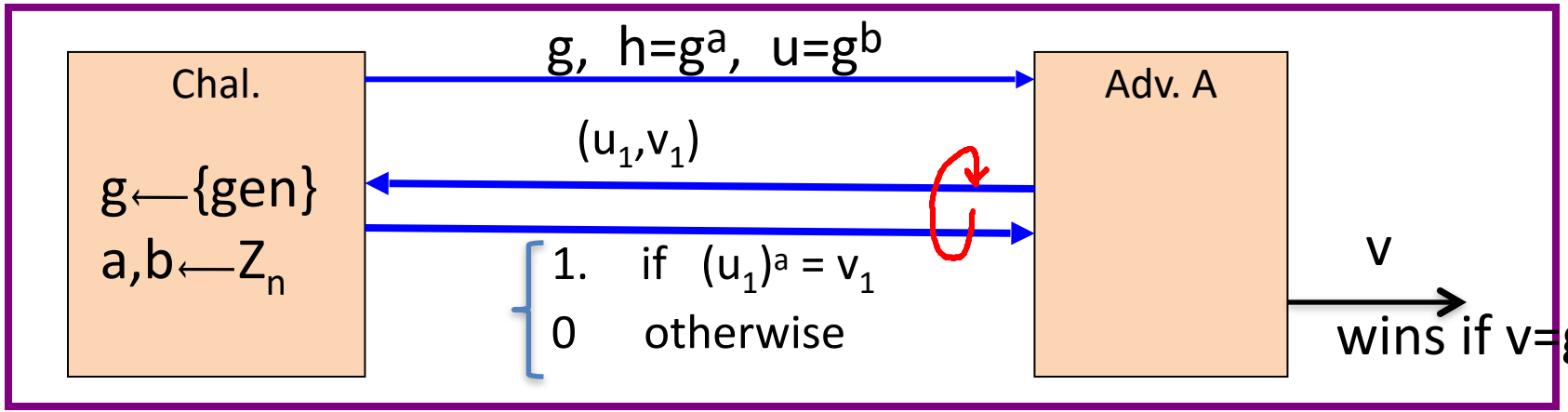
Questions: (1) can we prove CCA security based on CDH?

(2) can we prove CCA security without random oracles?

ElGamal chosen ciphertext security?

To prove chosen ciphertext security need stronger assumption

Interactive Diffie-Hellman (IDH) in group G:



IDH holds in G if: **$\forall$efficient A: Pr[A outputs g^{ab}] < negligible**

Decisional Diffie-Hellman Assumption

Decisional Diffie-Hellman Assumption (DDHA):

Hard to distinguish between g^{xy} and a uniformly random group element, given g, g^x and g^y
That is, the following two distributions are computationally indistinguishable:

$$(g, g^x, g^y, g^{xy}) \approx (g, g^x, g^y, u)$$

DH/El Gamal is IND-secure under the DDH assumption on the given group.

DLOG: more generally

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

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

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 candidates:

(1) $(Z_p)^*$ for large p, (2) Elliptic curve groups mod p

Computing Dlog in $(Z_p)^*$ (n-bit prime p)

Best known algorithm (GNFS): run time exp($\tilde{O}(\sqrt[3]{n})$)

cipher key size	modulus size	Elliptic Curve group size
80 bits	1024 bits	160 bits
128 bits	3072 bits	256 bits
256 bits (AES)	**15360** bits	512 bits

As a result: slow transition away from (mod p) to elliptic curves

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 generators g, h of G

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

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

Proof: Suppose we are given a collision $H(x_0,y_0) = H(x_1,y_1)$

$\neq 0$

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

Further reading

- A Computational Introduction to Number Theory and Algebra,
  V. Shoup, 2008 (V2), Chapter 1-4, 11, 12

  Available at **//shoup.net/ntb/ntb-v2.pdf**