# The ElGamal Public-key System

Recap:

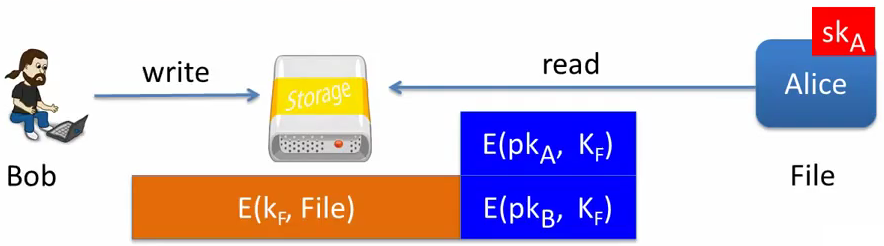
Public key encryption: (Gen, E, D)

## Public-key encryption applications

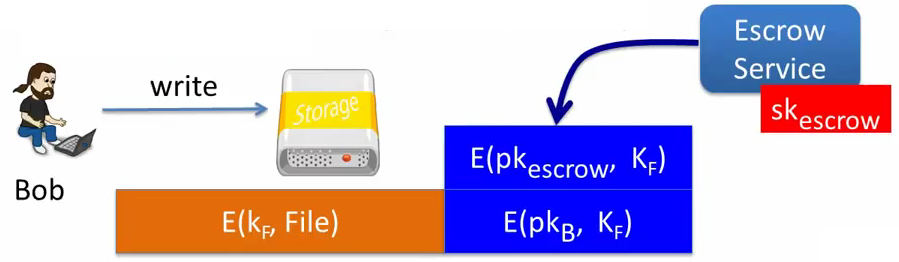
Key exchange (e.g. in HTTPS)

Encryption in non-interactive settings:

* Secure Email: Bob has Alice’s pub-key and sends her an email
* Encrypted File Systems



* Key escrow: data recovery without Bob’s key:



## Constructions

* Previous lecture: based on trapdoor functions (such as RSA)

- Schemes: ISO standard, OAEP+, …

* This lecture: based on the Diffie-Hellman protocol

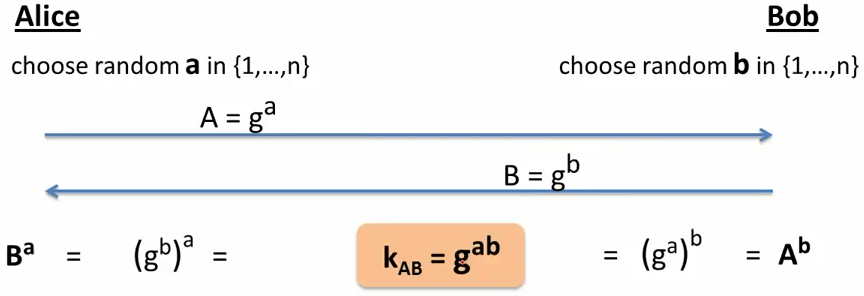
- Schemes: ElGamal encryption and variants (e.g. used in GPG)

Security goals: chosen ciphertext security

## Review: the Diffie-Hellman protocol (1997)

Fix a finite cyclic group G (e.g. G=(Zp)\* ) of orden n

Fix a generator g in G (i.e. G={1,g,g2,g3,..,gn-1} ) (gn=1)



ElGamal: converting to pub-key encryption (1984)

Fix a finite cyclic group G (e.g. G=(Zp)\* ) of orden n

Fix a generator g in G (i.e. G={1,g,g2,g3,..,gn-1} ) (gn=1)

ct = [ B=gb, encrypt message m with k ]

k is a symmetric key derived from Ab = gab

To decrypt: compute gab = Ba, derive k and decrypt ct

## The ElGamal system (a modern view)

G: finite cyclic group of order n

(ES, DS): symmetric authenticated encryption defined over (K,M,C)

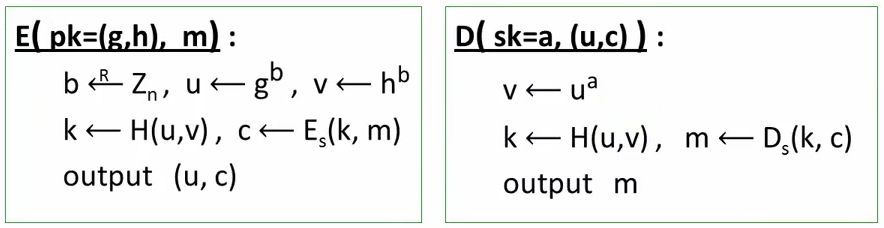
H: G2→K a hash function

We construct a pub-key encryption system (Gen, E, D):

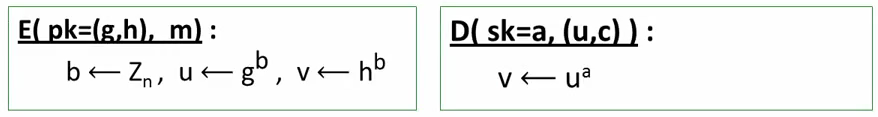
Key generation Gen:

- choose random generator g in G and random a in ZN

- output sk=a , pk = (g, h=ga)



## ElGamal performance



**Encryption**: 2 exp (fixed basis)

- Can pre-compute [ g(2^i), h(2^i) for i=1,..,log2n ]

- 3x speed-up (or more)

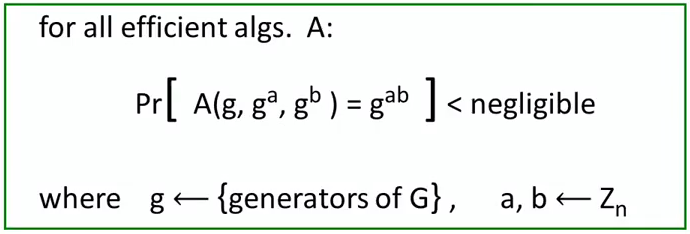
**Decryption**: 1 exp (variable basis)

# ElGamal security

## Computational Diffie-Hellman Assumption

G: finite cyclic group of orden n

Computational DH (CDH) assumption holds in G if: g, ga, gb =/=> gab

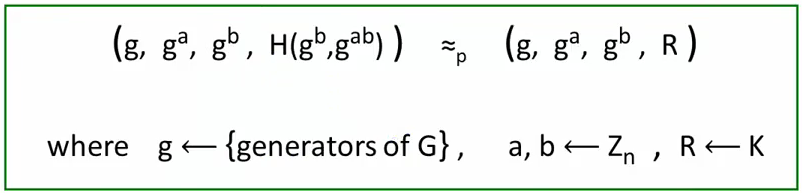


This assumption is not ideal for analysing the security of ElGamal system and instead we will make a stronger assumption:

## Hash Diffie-Hellman assumption

G: Finite cyclic group of order n, H: G2 → K a hash function

Def: Hash-DH (HDH) assumption holds for (G,H) if:



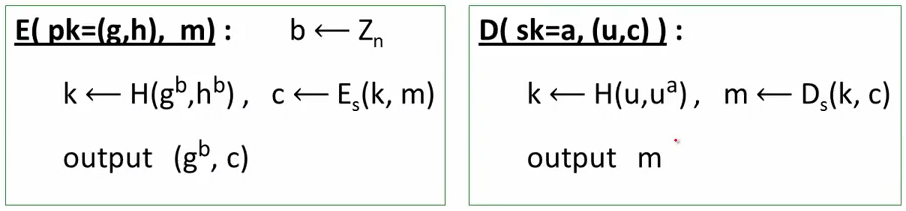
We will demonstrate that if CDH is easy in G ⇒ HDH is easy in (G,H) for all H, | Im(H) | >= 2

That almost evident because if CDH is easy then given g, ga, and gb I can calculate gab. Then I can calculate H(gb, gab) by myself and then recognize where the tuple came from, if from H(gb, gab) or if it is a random value R.

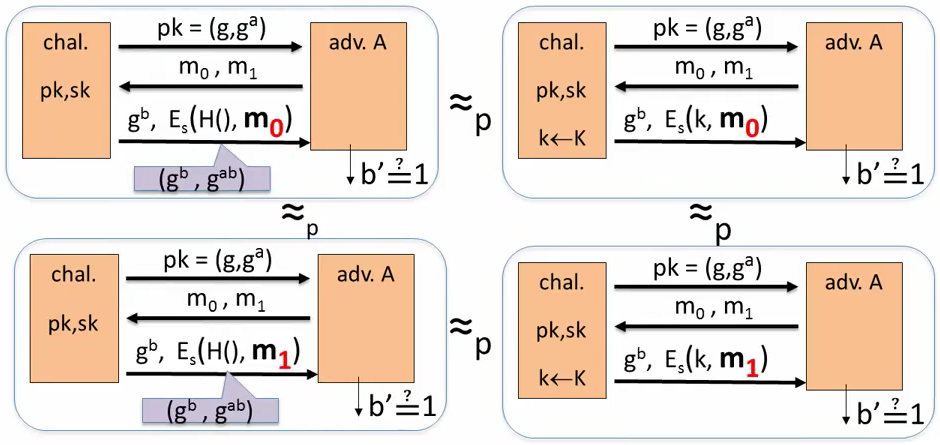
## ElGamal is semantically secure under Hash-DH

KeyGen: g ← {generators of G}, a ← Zn

output pk=(g,h=ga), sk=a



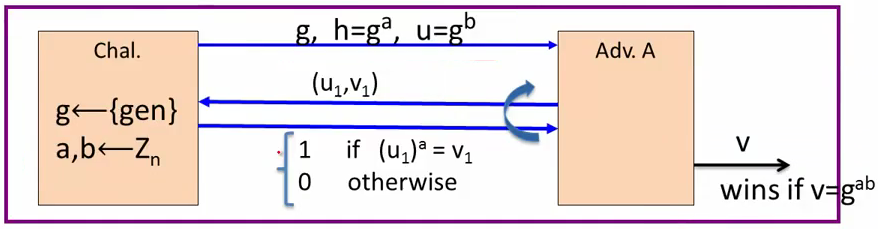
## ElGamal is semantically secure under Hash-DH



## ElGamal chosen ciphertext security

To prove chosen ciphertext security need stronger assumption

Interactive Diffie-Hellman (IDH) in group G:



IDH holds in G if: for all efficient A: Pr[A outputs gab] < negligible

**Security Theorem:**

If **IDH** holds in the group G, **(Es, Ds)** provides authenticated encryption

and **H**: G2→K is a “random oracle”

then **ElGamal** is CCAro secure.

Questions:

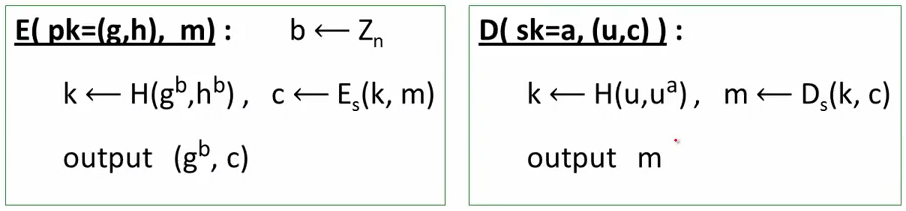
1.- Can we prove CCA security based on CDH?

2.- can we prove CCA security without random oracles?

# Variants of ElGamal with Better Security Analysis

KeyGen: g← {generators of G} , a ← Zn

output pk=(g,h=ga) , sk=a



## ElGamal chosen ciphertext security

Security Theorem:

If IDH holds in the group G, (ES, DS) provides authenticated encryption and H: G2 → K is a “random oracle” then ElGamal is CCAro secure.

Can we prove CCA security based on CDH (g,ga,gb -/-> gab) ?

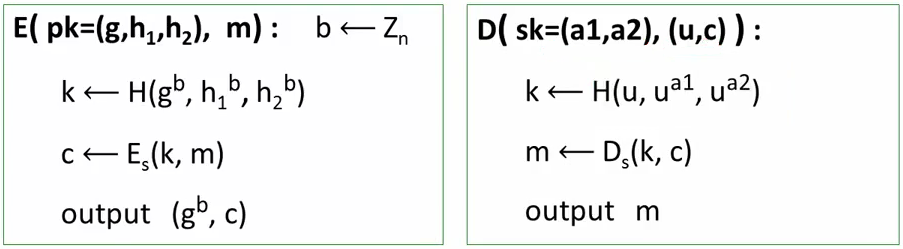
Option 1: use group G where CDH = IDH (a.k.a. bilinear group)

Option 2: change the ElGamal system

## Variants: twin ElGamal [CKS’08]

**KeyGen**: g ← {generators of G} , a1, a2 ← Zn

output pk = (g, h1=ga1, h2=ga2) , sk = (a1, a2)



## Chosen ciphertext security

**Security Theorem:**

If CDH holds in the group G, (ES,DS) provides authenticated encryption and H: G3→K is a “random oracle” then twin ElGamal is CCAro secure.

Cost: one more exponentiation during encryption/decryption

- Is it worth it? No one knows…

## ElGamal security without random oracles?

Can we prove CCA security without random oracles?

Option 1: use Hash-DH assumption in “bilinear groups”

- Special elliptic curve with more structure [CHK’04 + BB’04]

Option 2: use Decision-DH assumption in any group [CS’98]

## Further reading

* The Decision Diffiel-Hellman problem. D. Boneh, ANTS 3, 1998
* Universal hash proofs and a paradigm for chosen ciphertext secure public key encryption. R. Cramer and V. Shoup, Eurocrypt 2002
* Chosen-ciphertext security from Identity-Based Encryption. D. Boneg, R. Canetti, S. Halevi, and J. Katz, SICOMP 2007
* The Twin Diffie-Hellman problem and applications. D. Cash, E. Kiltz, V. Shoup, Eurocrypt 2008
* Efficient chosen-ciphertext security via extractable hash proofs. H. Wee, Crypto 2010

# A Unifying Theme

## One-way functions (informal)

A function f: X → Y is one-way if

* There is an efficient algorithm to evaluate f(·), but
* Inverting f is hard:

for all efficient A and x ← X :

Pr[ f( A(f(x)) ) = f(x) ] < negligible

Functions tha are not one-way: f(x) = x, f(x) = 0

## Example 1: generic one-way functions

Let f: X → Y be a secure PRG (where |Y| >> |X| ) (e.g. f built using det. counter mode)

**Lemma**: f a secure PRG ⇒ f is one-way

Proof sketch:

A invert f ⇒ B(y) = F(A(y)) = y output 0

output 1 otherwise

is a distinguisher

Generic: no special properties. Difficult to use for key exchange.

## Example 2: The DLOG one-way function

Fix a finite cyclic group G (e.g. G = (Zp)\* ) of order n

g: a random generator in G (i.e. G={1,g,g2,g3,..., gn-1} )

**Define**: f: Zn → G as F(x) = gx in G

**Lemma**: Dlog hard in G ⇒ f is one-way

**Properties**: f(x), f(y) ⇒ f(x+y) = f(x) \* f(y) ⇒ Key-exchange and public-key encryption

## Example 3: The RSA one-way function

Choose random primes p,q = 1024 bits. Set N=pq

Choose integers e,d s.t. e·d = 1 (mod phi(N))

**Define**: f:ZN\* → ZN\* as f(x) = xe in ZN

**Lemma**: f in one-way under the RSA assumption

**Properties**: f(x·y) = f(x) · f(y) and f has a trapdoor

## Summary

Public key encryption:

Made possible by one-way functions with special properties

Homomorphic properties and trapdoors

f(x),f(y) ⇒ f(x+y) or f(x·y)