

# Course Business

- **Homework 4 Released**
- Bonus Problem (10 Points)
  - Second bonus problem (5 pts) is easiest to solve with Mathematica
  - <https://sandbox.open.wolframcloud.com>
- **Homework 3 Grades**

<b>Minimum Value</b>	<b>60.00</b>
<b>Maximum Value</b>	<b>93.00</b>
<b>Range</b>	<b>33.00</b>
<b>Average</b>	<b>83.10</b>
<b>Median</b>	<b>87.00</b>
<b>Standard Deviation</b>	<b>9.00</b>
<b>Variance</b>	<b>81.09</b>

# Cryptography

## CS 555

### **Week 12:**

- Discrete Log Attacks + NIST Recommendations for Concrete Security Parameters
- Key Management
- Formalizing Public Key Encryption
- El Gamal

**Readings:** Katz and Lindell Chapter 10 & Chapter 11.1-11.2, 11.4

Week 12: Topic 0: Discrete Log  
Attacks + NIST  
Recommendations for Concrete  
Security Parameters

# Factoring Algorithms (Summary)

- Pollard's p-1 Algorithm
  - Works when  $N = pq$  where  $(p-1)$  has only "small" prime factors
  - **Defense:** Ensure that  $p$  (resp.  $q$ ) is a strong prime ( $(p-1)$  has no "small" prime factors).
  - **Note:** A random prime is strong with high probability.
- Pollard's Rho Algorithm
  - General purpose factoring algorithm
  - **Core:** Low Space Cycle Detection
  - **Time:**  $T(N) = O(\sqrt[4]{N} \text{ polylog}(N))$
  - Naïve Algorithm takes time  $O(\sqrt{N} \text{ polylog}(N))$  to factor
- Quadratic Sieve
  - **Time:**  $2^{O(\sqrt{\log N \log \log N})} = 2^{O(\sqrt{n \log n})}$  (sub-exponential, but not polynomial time)
  - Preprocessing + Linear Algebra: find  $x, y \in \mathbb{Z}_N^*$  such that  $x^2 = y^2 \pmod{N}$  and  $x \not\equiv \pm y \pmod{N}$ ?

# Discrete Log Attacks

- Pohlig-Hellman Algorithm
  - Given a cyclic group  $\mathbb{G}$  of non-prime order  $q = |\mathbb{G}| = rp$
  - Reduce discrete log problem to discrete problem(s) for subgroup(s) of order  $p$  (or smaller).
  - Preference for prime order subgroups in cryptography
- Baby-step/Giant-Step Algorithm
  - Solve discrete logarithm in time  $O(\sqrt{q} \text{ polylog}(q))$
- Pollard's Rho Algorithm
  - Solve discrete logarithm in time  $O(\sqrt{q} \text{ polylog}(q))$
  - Bonus: Constant memory!
- Index Calculus Algorithm
  - Similar to quadratic sieve
  - Runs in sub-exponential time  $2^{O(\sqrt{\log q \log \log q})}$
  - Specific to the group  $\mathbb{Z}_p^*$  (e.g., attack doesn't work elliptic-curves)

# Discrete Log Attacks

- Baby-step/Giant-Step Algorithm
  - Solve discrete logarithm in time  $O(\sqrt{q} \text{polylog}(q))$
  - Requires memory  $O(\sqrt{q} \text{polylog}(q))$
- Pollard's Rho Algorithm
  - Solve discrete logarithm in time  $O(\sqrt{q} \text{polylog}(q))$
  - Bonus: Constant memory!
- **Key Idea:** Low-Space Birthday Attack (\*) using our collision resistant hash function

$$\begin{aligned} H_{g,h}(x_1, x_2) &= g^{x_1} h^{x_2} \\ H_{g,h}(y_1, y_2) &= H_{g,h}(x_1, x_2) \rightarrow h^{y_2 - x_2} = g^{x_1 - y_1} \\ &\rightarrow h = g^{(x_1 - y_1)(y_2 - x_2)^{-1}} \end{aligned}$$

(\*) A few small technical details to address

# Discrete Log Attacks

- Baby-step/Giant-Step Algorithm
  - Solve discrete logarithm in time  $O(\sqrt{q} \text{polylog}(q))$
  - Requires memory  $O(\sqrt{q} \text{polylog}(q))$
- Pollard's Rho Algorithm
  - Solve discrete logarithm in time  $O(\sqrt{q} \text{polylog}(q))$
  - Bonus: Constant memory!
- **Key Idea:** Low-Space Birthday Attack (\*)

$$H_{g,h}(x_1, x_2) = g^{x_1} h^{x_2}$$
$$H_{g,h}(y_1, y_2) = H_{g,h}(x_1, x_2)$$

$$\rightarrow h^{y_2 - x_2} = g^{x_1 - y_1}$$
$$\rightarrow h = g^{(x_1 - y_1)(y_2 - x_2)^{-1}}$$

(\*) A few small technical details to address

**Remark:** We used discrete-log problem to construct collision resistant hash functions.

Security Reduction showed that attack on collision resistant hash function yields attack on discrete log.

→ Generic attack on collision resistant hash functions (e.g., low space birthday attack) yields generic attack on discrete log.

# Discrete Log Attacks

- Index Calculus Algorithm
  - Similar to quadratic sieve
  - Runs in sub-exponential time  $2^{O(\sqrt{\log p \log \log p})}$
  - Specific to the group  $\mathbb{Z}_p^*$  (e.g., attack doesn't work elliptic-curves)
- As before let  $\{p_1, \dots, p_k\}$  denote the set of prime numbers  $< B$ .
- **Step 1.A:** Find  $\ell > k$  distinct values  $x_1, \dots, x_k$  such that  $g_j = [g^{x_j} \bmod p]$  is B-smooth for each  $j$ . That is

$$g_j = \prod_{i=1}^k p_i^{e_{i,j}}.$$



# Discrete Log Attacks

- As before let  $\{p_1, \dots, p_k\}$  be set of prime numbers  $< B$ .
- **Step 1.A:** Find  $\ell > k$  distinct values  $x_1, \dots, x_k$  such that  $g_j = [g^{x_j} \bmod p]$  is B-smooth for each  $j$ . That is

$$g_j = \prod_{i=1}^k p_i^{e_{i,j}}.$$

- **Step 1.B:** Use linear algebra to solve the equations

$$x_j = \sum_{i=1}^k (\mathbf{log}_g \mathbf{p}_i) \times e_{i,j} \bmod (p - 1).$$

(Note: the  $\mathbf{log}_g \mathbf{p}_i$ 's are the unknowns)

# Discrete Log

- As before let  $\{p_1, \dots, p_k\}$  be set of prime numbers  $< B$ .
- **Step 1 (precomputation):** Obtain  $y_1, \dots, y_k$  such that  $p_i = g^{y_i} \text{ mod } p$ .
- **Step 2:** Given discrete log challenge  $h = g^x \text{ mod } p$ .
  - Find  $z$  such that  $[g^z h \text{ mod } p]$  is  $B$ -smooth

$$\begin{aligned} [g^z h \text{ mod } p] &= \prod_{i=1}^k p_i^{e_i} \\ &= \prod_{i=1}^k (g^{y_i})^{e_i} = g^{\sum_i e_i y_i} \end{aligned}$$

# Discrete Log

- As before let  $\{p_1, \dots, p_k\}$  be set of prime numbers  $< B$ .
- **Step 1 (precomputation):** Obtain  $y_1, \dots, y_k$  such that  $p_i = g^{y_i} \text{ mod } p$ .
- **Step 2:** Given discrete log challenge  $h = g^x \text{ mod } p$ .

- Find  $z$  such that  $[g^z h \text{ mod } p]$  is  $B$ -smooth

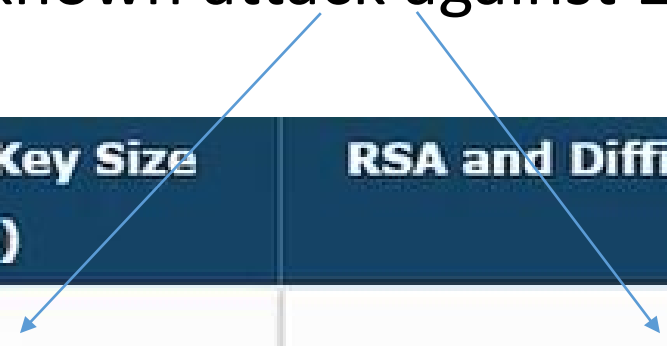
$$[g^z h \text{ mod } p] = g^{\sum_i e_i y_i} \rightarrow h = g^{\sum_i e_i y_i - z}$$

$$\rightarrow x = \sum_i e_i y_i - z$$

- **Remark:** Precomputation costs can be amortized over many discrete log instances
  - In practice, the same group  $\mathbb{G} = \langle g \rangle$  and generator  $g$  are used repeatedly.

# NIST Guidelines (Concrete Security)

Best known attack against 1024 bit RSA takes time (approximately)  $2^{80}$



<b>Symmetric Key Size (bits)</b>	<b>RSA and Diffie-Hellman Key Size (bits)</b>	<b>Elliptic Curve Key Size (bits)</b>
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Table 1: NIST Recommended Key Sizes

# NIST Guidelines (Concrete Security)

Diffie-Hellman uses subgroup of  $\mathbb{Z}_p^*$  size  $q$

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Table 1: NIST Recommended Key Sizes

# NIST Guidelines (Concrete Security)

$$112 \text{ bits} = \frac{\log 2^{224}}{2} = \log \sqrt{2^{224}} \text{ bits (Pollard's Rho)}$$

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Table 1: NIST Recommended Key Sizes

# NIST Guidelines (Concrete Security)

112 bits  $\approx \sqrt{2048 \log 2048}$  bits (Index Calculus)

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)		Elliptic Curve Key Size (bits)
80	1024		160
112	2048	q=224 bits	224
128	3072	q=256 bits	256
192	7680	q=384 bits	384
256	15360	q=512 bits	521

Table 1: NIST Recommended Key Sizes

Security Strength		2011 through 2013	2014 through 2030	2031 and Beyond
80	Applying	Deprecated	Disallowed	
	Processing	Legacy use		
112	Applying	Acceptable	Acceptable	Disallowed
	Processing			Legacy use
128	Applying/Processing	Acceptable	Acceptable	Acceptable
192		Acceptable	Acceptable	Acceptable
256		Acceptable	Acceptable	Acceptable

NIST's security strength guidelines, from Specialist Publication SP 800-57  
*Recommendation for Key Management – Part 1: General (Revision 3)*



# Week 12: Topic 1: Key Management



# Key-Exchange Problem

- **Key-Exchange Problem:**
  - Obi-Wan and Yoda want to communicate securely
  - Suppose that
    - Obi-Wan and Yoda don't have time to meet privately and generate one
    - Obi-Wan and Yoda share a symmetric key with Anakin
    - Suppose that they fully trust Anakin



# Key-Distribution Center (with Symmetric Key-Crypto)



$K_{\text{obiwan}}$ : Shared key between Obiwan and Anakin

$\text{Enc}(K_{\text{obiwan}}, \text{"I would like to talk to Yoda"})$

Ok, here is a fresh key that no sith lord has seen

$c_1 = \text{Enc}(K_{\text{obiwan}}, \text{ts}, K_{\text{new}}),$   
 $c_2 = \text{Enc}(K_{\text{yoda}}, \text{ts}, \text{"Obiwan/Yoda"}, K_{\text{new}})$



$K_{\text{yoda}}$ : Shared key between yoda and Anakin

# Key-Distribution Center (with Symmetric Key-Crypto)



$\text{Enc}(K_{\text{obiwan}}, \text{"I would like to talk to Yoda"})$

Ok, here is a fresh key that no sith lord has seen



$c_1 = \text{Enc}(K_{\text{obiwan}}, ts, K_{\text{new}})$ ,

$c_2 = \text{Enc}(K_{\text{yoda}}, ts, \text{"Obiwan/Yoda"}, K_{\text{new}})$

$\text{Enc}(K_{\text{new}}, \text{"Its me, Obiwan, let's talk"})$   
 $c_2 = \text{Enc}(K_{\text{yoda}}, ts, \text{"Obiwan/Yoda"}, K_{\text{new}})$



# Key-Distribution Center (with Symmetric Key-Crypto)

- **Vulnerability:** If Key-Distribution Center is compromised then **all** security guarantees are broken.
  - KDC is a valuable target for attackers
  - Possibility of insider attacks (e.g., employees)



- **Denial of Service (DOS) Attack:** If KDC is down then secure communication is temporarily impossible.

# Key-Distribution Center (with Symmetric Key-Crypto)

- **Benefit:** Authenticated Encryption provides authentication as well
  - Yoda can be sure he is talking to Obiwan (assuming he trusts the KDC)
- Kerberos uses similar protocol
  - Yoda's key and Obiwan's key are typically derived from a password that they know.
  - **Vulnerability:** An eavesdropping attacker can mount a brute-force attack on the (low-entropy) passwords to recover  $K_{yoda}$  and  $K_{obiwan}$ .
- **Recommendation:** Always use Public Key Initialization with Kerberos

# Key-Explosion Problem

- To avoid use a trusted KDC we could have every pair of users exchange private keys
- How many private keys per person?
  - **Answer:**  $n-1$
  - Need to meet up with  $n-1$  different users in person!
- Key Explosion Problem
  - $n$  can get very big if you are Google or Amazon!





# Diffie-Hellman Key Exchange

1. Alice picks  $x_A$  and sends  $g^{x_A}$  to Bob
2. Bob picks  $x_B$  and sends  $g^{x_B}$  to Alice
3. Alice and Bob can both compute  $K_{A,B} = g^{x_B x_A}$

# Key-Exchange Experiment $KE_{A,\Pi}^{eav}(n)$ :

- Two parties run  $\Pi$  to exchange secret messages (with security parameter  $1^n$ ).
- Let **trans** be a transcript which contains all messages sent and let  $k$  be the secret key output by each party.
- Let  $b$  be a random bit and let  $\mathbf{k}_b = k$  if  $b=0$ ; otherwise  $\mathbf{k}_b$  is sampled uniformly at random.
- Attacker  $A$  is given **trans** and  $\mathbf{k}_b$  (passive attacker).
- Attacker outputs  $b'$  ( $KE_{A,\Pi}^{eav}(n)=1$  if and only if  $b=b'$ )

Security of  $\Pi$  against an eavesdropping attacker: For all PPT  $A$  there is a negligible function **negl** such that

$$\Pr[KE_{A,\Pi}^{eav}(n)] = \frac{1}{2} + \mathbf{negl}(n).$$

# Diffie-Hellman Key-Exchange is Secure

**Theorem:** If the decisional Diffie-Hellman problem is hard relative to group generator  $\mathcal{G}$  then the Diffie-Hellman key-exchange protocol  $\Pi$  is secure in the presence of an eavesdropper (\*).

(\*) Assuming keys are chosen uniformly at random from the cyclic group  $\mathbb{G}$

## Protocol $\Pi$

1. Alice picks  $x_A$  and sends  $g^{x_A}$  to Bob
2. Bob picks  $x_B$  and sends  $g^{x_B}$  to Alice
3. Alice and Bob can both compute  $K_{A,B} = g^{x_B x_A}$

# Diffie-Hellman Assumptions

## Computational Diffie-Hellman Problem (CDH)

- Attacker is given  $h_1 = g^{x_1} \in \mathbb{G}$  and  $h_2 = g^{x_2} \in \mathbb{G}$ .
- Attacker's goal is to find  $g^{x_1 x_2} = (h_1)^{x_2} = (h_2)^{x_1}$
- **CDH Assumption:** For all PPT A there is a negligible function  $\text{negl}$  upper bounding the probability that A succeeds

## Decisional Diffie-Hellman Problem (DDH)

- Let  $z_0 = g^{x_1 x_2}$  and let  $z_1 = g^r$ , where  $x_1, x_2$  and  $r$  are random
- Attacker is given  $g^{x_1}, g^{x_2}$  and  $z_b$  (for a random bit  $b$ )
- Attacker's goal is to guess  $b$
- **DDH Assumption:** For all PPT A there is a negligible function  $\text{negl}$  such that A succeeds with probability at most  $\frac{1}{2} + \text{negl}(n)$ .

# Diffie-Hellman Key Exchange

1. Alice picks  $x_A$  and sends  $g^{x_A}$  to Bob
2. Bob picks  $x_B$  and sends  $g^{x_B}$  to Alice
3. Alice and Bob can both compute  $K_{A,B} = g^{x_B x_A}$

**Intuition:** Decisional Diffie-Hellman assumption implies that a passive attacker who observes  $g^{x_A}$  and  $g^{x_B}$  still cannot distinguish between  $K_{A,B} = g^{x_B x_A}$  and a random group element.

# Diffie-Hellman Key-Exchange is Secure

**Theorem:** If the decisional Diffie-Hellman problem is hard relative to group generator  $\mathcal{G}$  then the Diffie-Hellman key-exchange protocol  $\Pi$  is secure in the presence of an eavesdropper (\*).

**Proof:**

$$\begin{aligned} & \Pr[KE_{A,\Pi}^{eav}(n) = 1] \\ &= \frac{1}{2}\Pr[KE_{A,\Pi}^{eav}(n) = 1 | b = 1] + \frac{1}{2}\Pr[KE_{A,\Pi}^{eav}(n) = 1 | b = 0] \\ &= \frac{1}{2}\Pr[A(\mathbb{G}, g, q, g^x, g^y, g^{xy}) = 1] + \frac{1}{2}\Pr[A(\mathbb{G}, g, q, g^x, g^y, g^z) = 1] \\ &= \frac{1}{2} + \frac{1}{2}(\Pr[A(\mathbb{G}, g, q, g^x, g^y, g^{xy}) = 1] - \Pr[A(\mathbb{G}, g, q, g^x, g^y, g^z) = 1]). \\ & \leq \frac{1}{2} + \frac{1}{2}\text{negl}(n) \text{ (by DDH)} \end{aligned}$$

(\*) Assuming keys are chosen uniformly at random from the cyclic group  $\mathbb{G}$

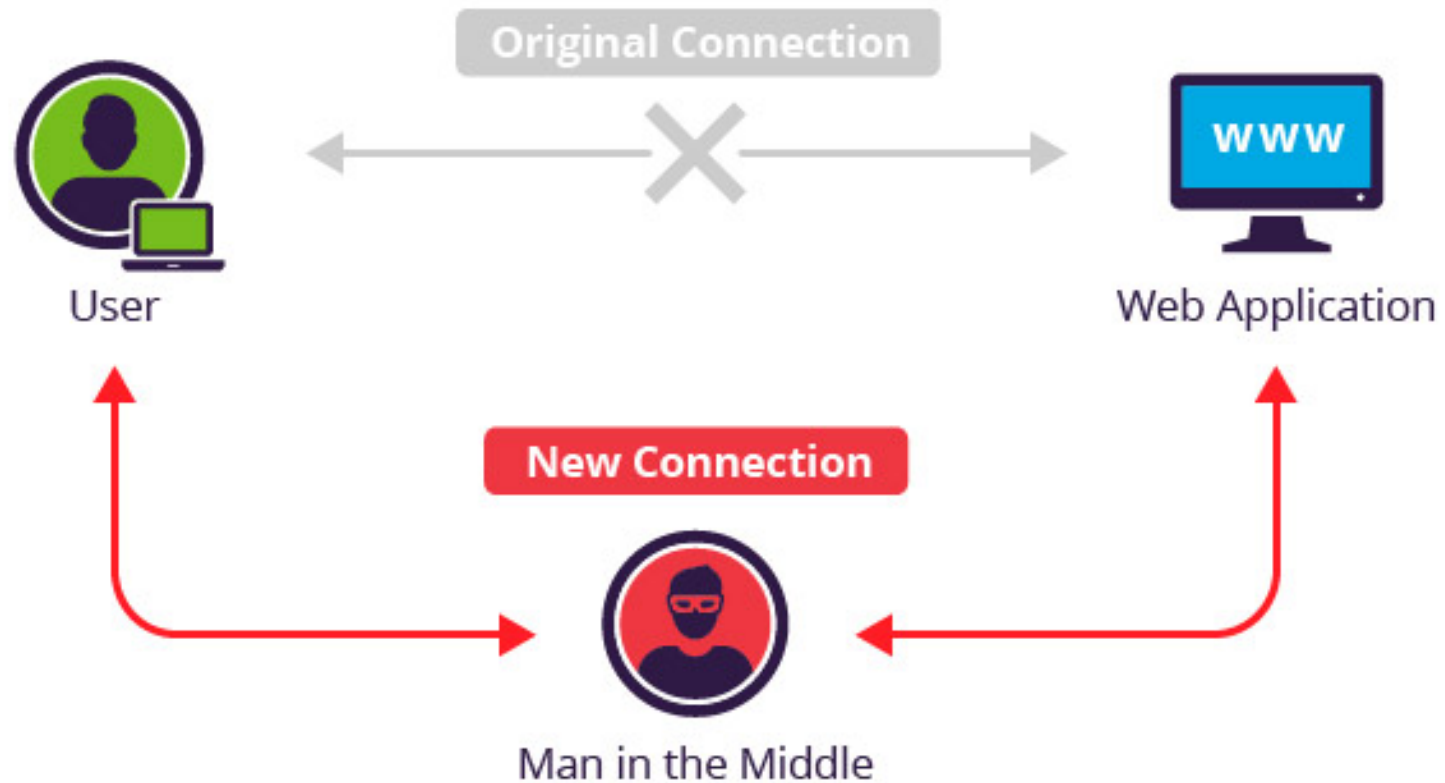
# Diffie-Hellman Key Exchange

1. Alice picks  $x_A$  and sends  $g^{x_A}$  to Bob
2. Bob picks  $x_B$  and sends  $g^{x_B}$  to Alice
3. Alice and Bob can both compute  $K_{A,B} = g^{x_B x_A}$

**Intuition:** Decisional Diffie-Hellman assumption implies that a passive attacker who observes  $g^{x_A}$  and  $g^{x_B}$  still cannot distinguish between  $K_{A,B} = g^{x_B x_A}$  and a random group element.

**Remark:** The protocol is vulnerable against active attackers who can tamper with messages.

# Man in the Middle Attack (MITM)





# Man in the Middle Attack (MITM)

1. Alice picks  $x_A$  and sends  $g^{x_A}$  to Bob
  - Eve intercepts  $g^{x_A}$ , picks  $x_E$  and sends  $g^{x_E}$  to Bob instead
2. Bob picks  $x_B$  and sends  $g^{x_B}$  to Alice
  1. Eve intercepts  $g^{x_B}$ , picks  $x_{E'}$  and sends  $g^{x_{E'}}$  to Alice instead
3. Eve computes  $g^{x_{E'}x_A}$  and  $g^{x_{E'}x_B}$ 
  1. Alice computes secret key  $g^{x_{E'}x_A}$  (shared with Eve not Bob)
  2. Bob computes  $g^{x_{E'}x_B}$  (shared with Eve not Alice)
4. Eve forwards messages between Alice and Bob (tampering with the messages if desired)
5. Neither Alice nor Bob can detect the attack

# Password Authenticated Key-Exchange

- Suppose Alice and Bob share a low-entropy password  $\text{pwd}$  and wish to communicate securely
  - (without using any trusted party)
  - Assuming an active attacker may try to mount a man-in-the-middle attack
- Can they do it?

## Tempting Approach:

- Alice and Bob both compute  $K = \text{KDF}(\text{pwd}) = H^n(\text{pwd})$  and communicate with using an authenticated encryption scheme.
- **Midterm Exam:** Secure in random oracle model if attacker cannot query random oracle too many time.

# Password Authenticated Key-Exchange

## Tempting Approach:

- Alice and Bob both compute  $K = \text{KDF}(\text{pwd}) = H^n(\text{pwd})$  and communicate with using an authenticated encryption scheme.
- **Midterm Exam:** Secure in random oracle model if attacker cannot query random oracle too many time.
- **Problems:**
  - In practice the attacker can (and will) query the random oracle many times.
  - In practice people tend to pick very weak passwords
  - Brute-force attack: Attacker enumerates over a dictionary of passwords and attempts to decrypt messages with  $K_{\text{pwd}'} = \text{KDF}(\text{pwd}')$  (only succeeds if  $K_{\text{pwd}'} = K$ ).
  - An offline attack (brute-force) will almost always succeed

# Password Authenticated Key-Exchange (PAKE)

## Better Approach (PAKE):

1. Alice and Bob both compute  $W = g^{pwd}$
2. Alice picks  $x_A$  and sends "Alice",  $X = g^{x_A}$  to Bob
3. Bob picks  $x_B$ , computes  $r = H(1, Alice, Bob, X)$  and  $Y = (X \times (W)^r)^{x_B}$  and sends Alice the following message: "Bob",  $Y$
4. Alice computes  $K = Y^z = g^{x_B}$  where  $z = 1/((pwd \times r) + x_A) \text{ mod } p$ . Alice sends the message  $V_A = H(2, Alice, Bob, X, Y, K)$  to Bob.
5. Bob verifies that  $V_A = H(2, Alice, Bob, X, Y, K)$  where  $K = g^{x_B}$ . Bob generates  $V_B = H(3, Alice, Bob, X, Y, K)$  and sends  $V_B$  to Alice.
6. Alice verifies that  $V_B = H(3, Alice, Bob, X, Y, Y^z)$  where  $z = 1/((pwd \times r) + x_A)$ .
7. If Alice and Bob don't terminate the session key is  $H(4, Alice, Bob, X, Y, K)$

## Security:

- No offline attack (brute-force) is possible. Attacker gets one password guess per instantiation of the protocol.
- If attacker is incorrect and he tampers with messages then he will cause the Alice & Bob to quit.
- If Alice and Bob accept the secret key  $K$  and the attacker did not know/guess the password then  $K$  is "just as good" as a truly random secret key.

# Key-Explosion Problem

- So far neither Diffie-Hellman Key Exchange nor PAKEs completely solved the problem
- PAKEs require a shared password
  - (n-1) shared passwords?
- Diffie-Hellman Key Exchange is vulnerable to man-in-the-middle
- Can use KDC to store database of public-keys (e.g.,  $g^{x_A}$ ) for each party.
  - Breached KDC doesn't reveal secret keys



# Public Key Revolution

- Digital Signatures can help
  - Private-Key Analogue: MAC
  - Private Key required to produce signature for a message  $m$
  - Anyone with Public Key can verify the message
- An authority could sign the message “Alice’s public key is  $g^{x_A}$ ”
- Anyone could use the authority’s public key to validate Alice’s public key
- The authority does not actually need to store  $g^{x_A}$ .
- In fact, if Alice has signature then she can use this to prove her identity to Bob (and Bob doesn’t need to interact the authority)

# Week 12 Topic 2: Formalizing Public Key Cryptography

# Public Key Encryption: Basic Terminology

- Plaintext/Plaintext Space
  - A message  $m \in \mathcal{M}$
- Ciphertext  $c \in \mathcal{C}$
- **Public/Private Key Pair  $(pk, sk) \in \mathcal{K}$**



# Public Key Encryption Syntax

- Three Algorithms

- $\text{Gen}(1^n, R)$  (Key-generation algorithm)

- Input: Random Bits  $R$

- Output:  $(pk, sk) \in \mathcal{K}$

- $\text{Enc}_{pk}(m) \in \mathcal{C}$  (Encryption algorithm)

- $\text{Dec}_{sk}(c)$  (Decryption algorithm)

- Input: Secret key  $sk$  and a ciphertext  $c$

- Output: a plaintext message  $m \in \mathcal{M}$

Alice must run key generation algorithm in advance and publishes the public key:  $pk$

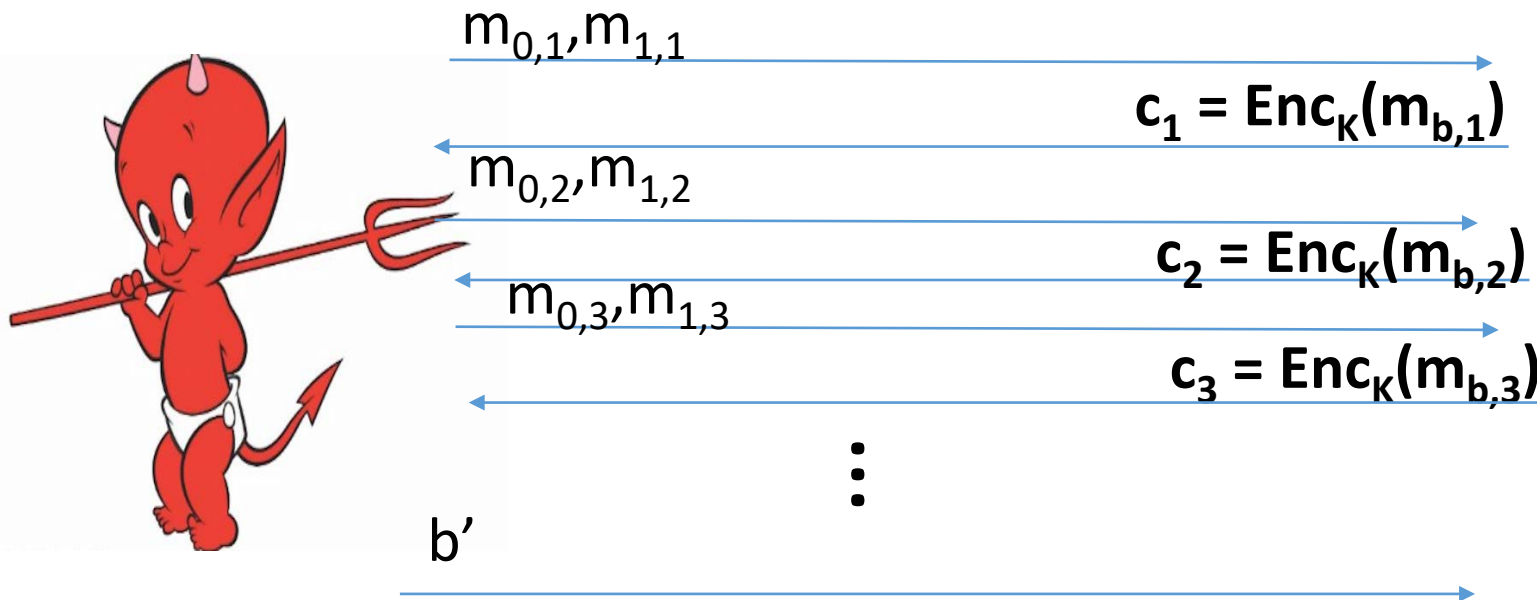
Assumption: Adversary only gets to see  $pk$  (not  $sk$ )

- **Invariant:**  $\text{Dec}_{sk}(\text{Enc}_{pk}(m))=m$

# Chosen-Plaintext Attacks

- Model ability of adversary to control or influence what the honest parties encrypt.
- Historical Example: Battle of Midway (WWII).
  - US Navy cryptanalysts were able to break Japanese code by tricking Japanese navy into encrypting a particular message
- Private Key Cryptography

# Recap CPA-Security (Symmetric Key Crypto)



Random bit  $b$   
 $K = \text{Gen}(\cdot)$



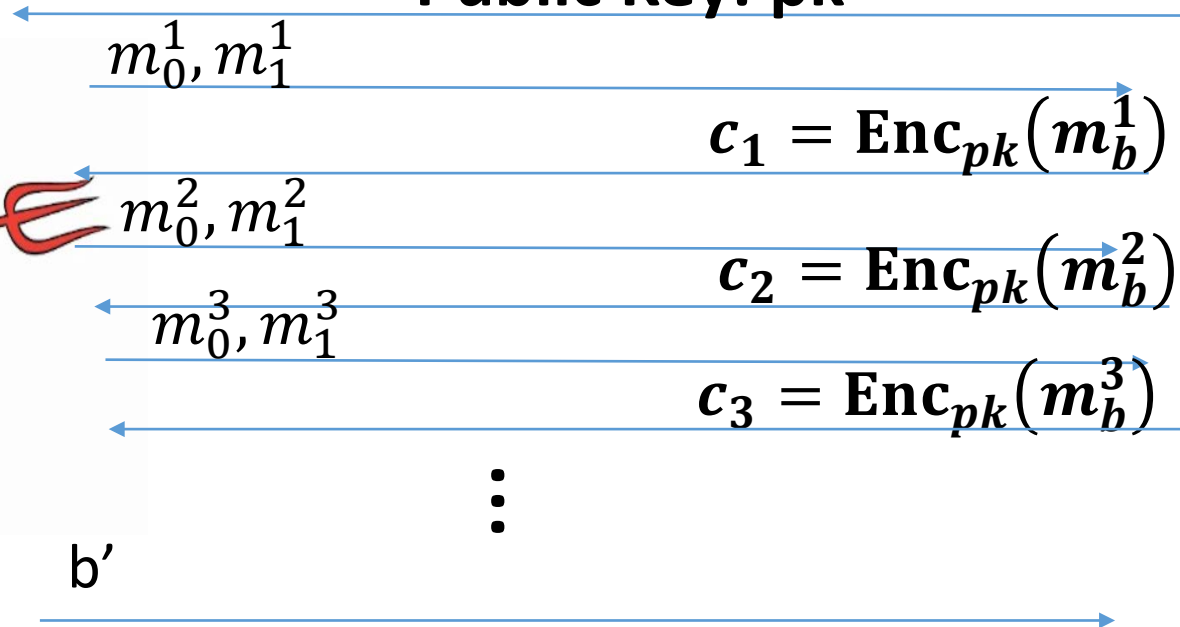
$$\forall PPT A \exists \mu \text{ (negligible) s. t.}$$
$$\Pr[A \text{ Guesses } b' = b] \leq \frac{1}{2} + \mu(n)$$

# Chosen-Plaintext Attacks

- Model ability of adversary to control or influence what the honest parties encrypt.
- Private Key Crypto
  - Attacker tricks victim into encrypting particular messages
- Public Key Cryptography
  - The attacker already has the public key  $pk$
  - Can encrypt any message s/he wants!
  - CPA Security is critical!

# CRA-Security ( $\text{PubK}_{A,\Pi}^{\text{LR-cpa}}(n)$ )

Public Key:  $pk$



Random bit  $b$   
 $(pk, sk) = \text{Gen}(\cdot)$



$$\forall PPT A \exists \mu \text{ (negligible) s.t.}$$

$$\Pr[\text{PubK}_{A,\Pi}^{\text{LR-cpa}}(n) = 1] \leq \frac{1}{2} + \mu(n)$$

# CPA-Security (Single Message)

*Formally, let  $\Pi = (Gen, Enc, Dec)$  denote the encryption scheme, call the experiment  $\text{PubK}_{A,\Pi}^{\text{LR-cpa}}(n)$  and define a random variable*

$$\text{PubK}_{A,\Pi}^{\text{LR-cpa}}(n) = 1 \quad \text{if } b = b'$$

$$\text{PubK}_{A,\Pi}^{\text{LR-cpa}}(n) = 0 \quad \text{otherwise}$$

*$\Pi$  has indistinguishable encryptions under a chosen plaintext attack if for all PPT adversaries  $A$ , there is a negligible function  $\mu$  such that*

$$\Pr[\text{PubK}_{A,\Pi}^{\text{LR-cpa}}(n) = 1] \leq \frac{1}{2} + \mu(n)$$

# Private Key Crypto

- CPA Security was stronger than eavesdropping security

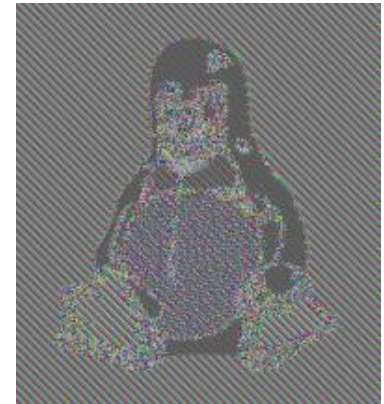
$$\text{Enc}_K(m) = G(K) \oplus m$$

Vs.

$$\text{Enc}_K(m) = \langle r, F_k(r) \oplus m \rangle$$

# Public Key Crypto

- **Fact 1: CPA Security and Eavesdropping Security are Equivalent**
  - Key Insight: The attacker has the public key so he doesn't gain anything from being able to query the encryption oracle!
- **Fact 2: Any deterministic encryption scheme is not CPA-Secure**
  - Historically overlooked in many real world public key crypto systems
- **Fact 3: Plain RSA is not CPA-Secure**
- **Fact 4: No Public Key Cryptosystem can achieve Perfect Secrecy!**
  - Exercise 11.1
  - Hint: Unbounded attacker can keep encrypting the message  $m$  using the public key to recover all possible encryptions of  $m$ .





# Encrypting Longer Messages

**Claim 11.7:** Let  $\Pi = (Gen, Enc, Dec)$  denote a CPA-Secure public key encryption scheme and let  $\Pi' = (Gen, Enc', Dec')$  be defined such that

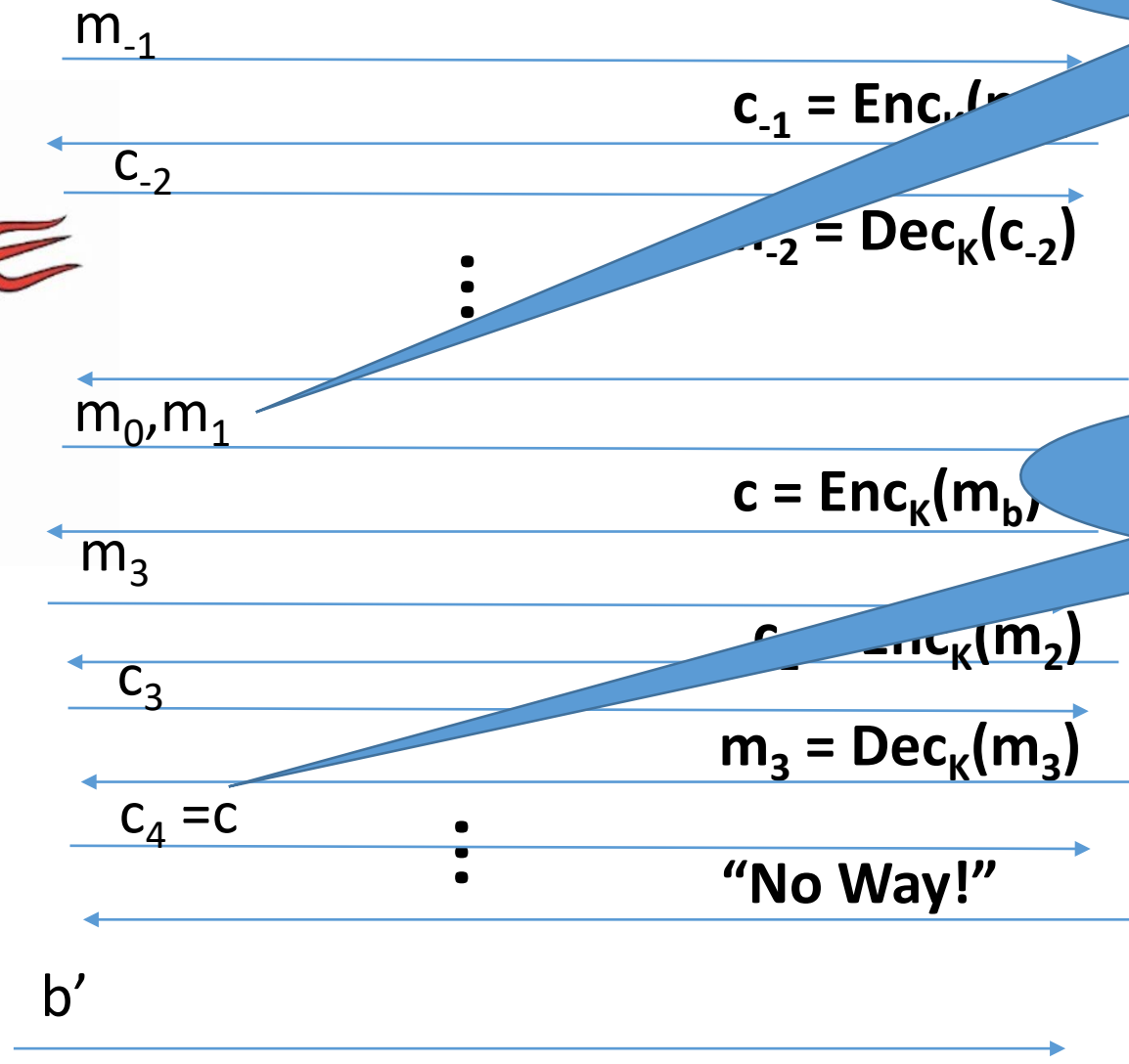
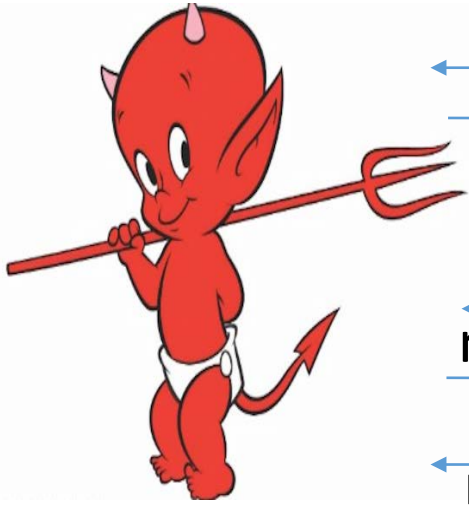
$$\mathbf{Enc}'_{pk}(m_1 \parallel m_2 \parallel \cdots \parallel m_\ell) = \mathbf{Enc}_{pk}(m_1) \parallel \cdots \parallel \mathbf{Enc}_{pk}(m_\ell)$$

Then  $\Pi'$  is also CPA-Secure.

# Chosen Ciphertext Attacks

- Models ability of attacker to obtain (partial) decryption of selected ciphertexts
- Attacker might intercept ciphertext  $c$  (sent from  $S$  to  $R$ ) and send  $c'$  instead.
  - After that attacker can observe receiver's behavior (abort, reply etc...)
- Attacker might send a modified ciphertext  $c'$  to receiver  $R$  in his own name.
  - E-mail response: Receiver might decrypt  $c'$  to obtain  $m'$  and include  $m'$  in the response to the attacker

# Recap CCA-Security (Symmetric)



We could set  $m_0 = m_{-1}$  or  $m_1 = m_{-2}$



However, we could still flip 1 bit of  $c$  and ask challenger to decrypt

Random bit  $b$   
 $K = \text{Gen}(\cdot)$



# Recap CCA-Security $\left(PrivK_{A,\Pi}^{cca}(n)\right)$

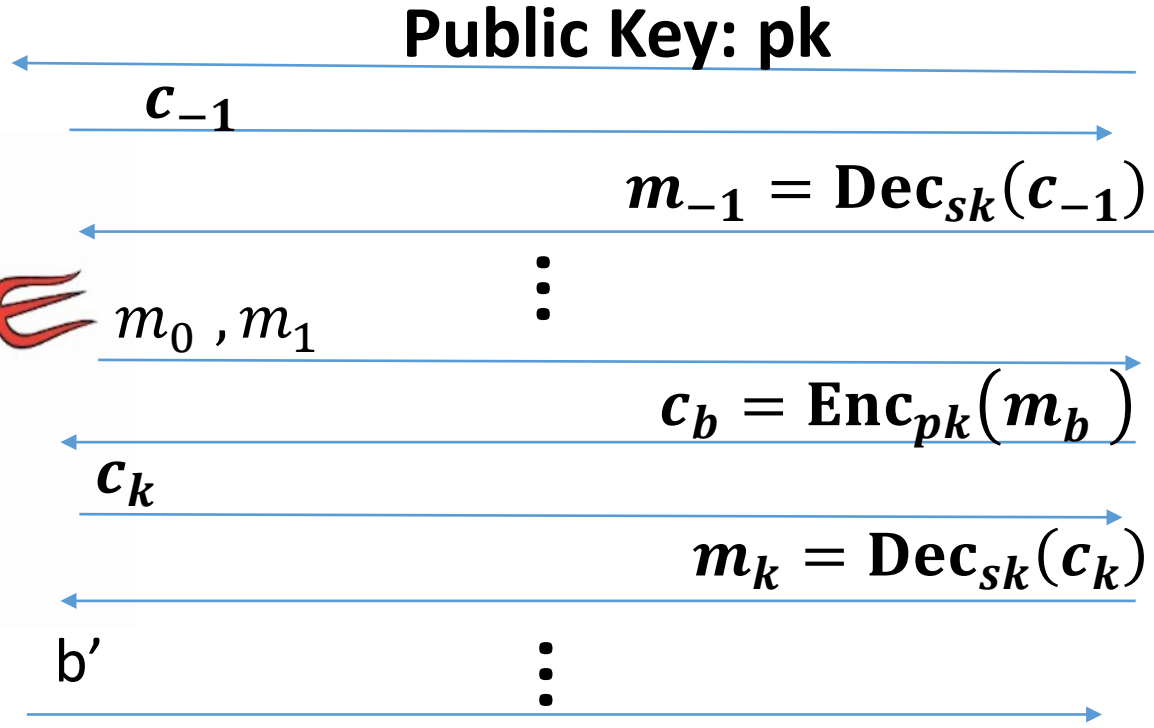
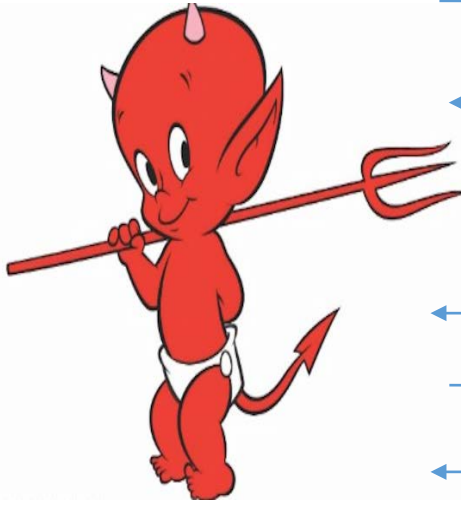
1. Challenger generates a secret key  $k$  and a bit  $b$
2. Adversary (A) is given oracle access to  $Enc_k$  and  $Dec_k$
3. Adversary outputs  $m_0, m_1$
4. Challenger sends the adversary  $c = Enc_k(m_b)$ .
5. Adversary maintains oracle access to  $Enc_k$  and  $Dec_k$ , however the adversary is not allowed to query  $Dec_k(c)$ .
6. Eventually, Adversary outputs  $b'$ .

$$PrivK_{A,\Pi}^{cca}(n) = 1 \text{ if } b = b'; \text{ otherwise } 0.$$

**CCA-Security:** For all PPT A exists a negligible function  $negl(n)$  s.t.

$$\Pr[PrivK_{A,\Pi}^{cca}(n) = 1] \leq \frac{1}{2} + negl(n)$$

# CCA-Security ( $\text{PubK}_{A,\Pi}^{\text{cca}}(n)$ )



Random bit  $b$   
 $(pk, sk) = \text{Gen}(\cdot)$



$$\forall PPT A \exists \mu \text{ (negligible) s.t.}$$

$$\Pr[\text{PubK}_{A,\Pi}^{\text{cca}}(n) = 1] \leq \frac{1}{2} + \mu(n)$$

# Encrypting Longer Messages

**Claim 11.7:** Let  $\Pi = (Gen, Enc, Dec)$  denote a CPA-Secure public key encryption scheme and let  $\Pi' = (Gen, Enc', Dec')$  be defined such that

$$\mathbf{Enc}'_{pk}(m_1 \parallel m_2 \parallel \cdots \parallel m_\ell) = \mathbf{Enc}_{pk}(m_1) \parallel \cdots \parallel \mathbf{Enc}_{pk}(m_\ell)$$

Then  $\Pi'$  is also CPA-Secure.

**Claim?** Let  $\Pi = (Gen, Enc, Dec)$  denote a CCA-Secure public key encryption scheme and let  $\Pi' = (Gen, Enc', Dec')$  be defined such that

$$\mathbf{Enc}'_{pk}(m_1 \parallel m_2 \parallel \cdots \parallel m_\ell) = \mathbf{Enc}_{pk}(m_1) \parallel \cdots \parallel \mathbf{Enc}_{pk}(m_\ell)$$

Then  $\Pi'$  is also CCA-Secure.

Is this second claim true?

# Encrypting Longer Messages

**Claim?** Let  $\Pi = (Gen, Enc, Dec)$  denote a **CCA**-Secure public key encryption scheme and let  $\Pi' = (Gen, Enc', Dec')$  be defined such that

$$\mathbf{Enc}'_{pk}(m_1 \parallel m_2 \parallel \cdots \parallel m_\ell) = \mathbf{Enc}_{pk}(m_1) \parallel \cdots \parallel \mathbf{Enc}_{pk}(m_\ell)$$

Then  $\Pi'$  is also **CCA**-Secure.

Is this second claim true?

**Answer: No!**

# Encrypting Longer Messages

**Fact:** Let  $\Pi = (Gen, Enc, Dec)$  denote a **CCA**-Secure public key encryption scheme and let  $\Pi' = (Gen, Enc', Dec')$  be defined such that

$$\mathbf{Enc}'_{pk}(m_1 \parallel m_2 \parallel \cdots \parallel m_\ell) = \mathbf{Enc}_{pk}(m_1) \parallel \cdots \parallel \mathbf{Enc}_{pk}(m_\ell)$$

Then  $\Pi'$  is **Provably Not CCA**-Secure.

1. Attacker sets  $m_0 = \mathbf{0}^n \parallel \mathbf{1}^n \parallel \mathbf{1}^n$  and  $m_1 = \mathbf{0}^n \parallel \mathbf{0}^n \parallel \mathbf{1}^n$  and gets  $c_b = \mathbf{Enc}'_{pk}(m_b) = c_{b,1} \parallel c_{b,2} \parallel c_{b,3}$
2. Attacker sets  $c' = c_{b,2} \parallel c_{b,3} \parallel c_{b,1}$ , queries the decryption oracle and gets

$$\mathbf{Dec}'_{sk}(c') = \begin{cases} \mathbf{1}^n \parallel \mathbf{1}^n \parallel \mathbf{0}^n & \text{if } b=0 \\ \mathbf{0}^n \parallel \mathbf{1}^n \parallel \mathbf{0}^n & \text{otherwise} \end{cases}$$



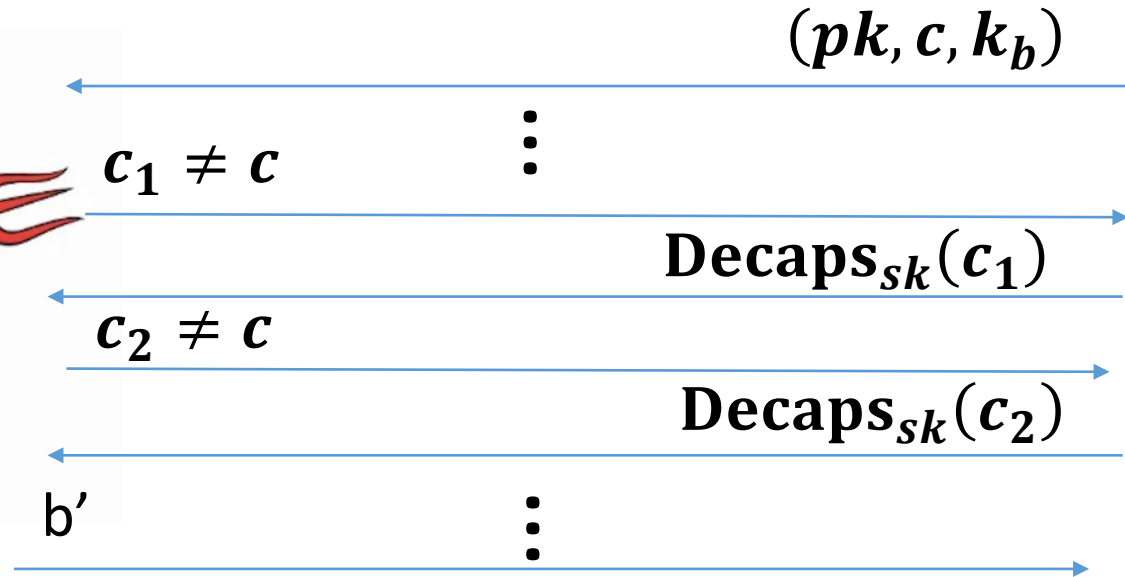
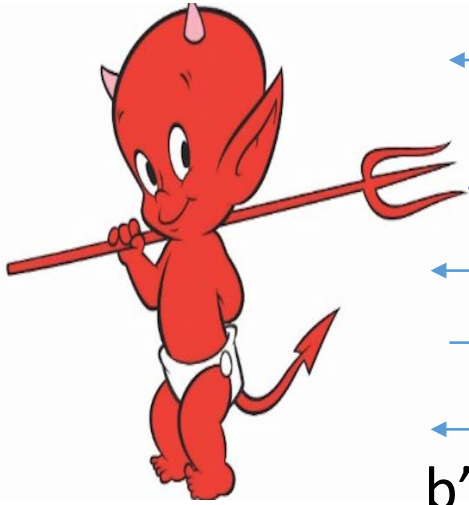
# Achieving CPA and CCA-Security

- Plain RSA is not CPA Secure (therefore, not CCA-Secure)
- El-Gamal (next class) is CPA-Secure, but not CCA-Secure
  - Homework 4
- Tools to build CCA-Secure Encryption
  - Key Encapsulation Mechanism

# Key Encapsulation Mechanism (KEM)

- Three Algorithms
  - $\text{Gen}(1^n, R)$  (Key-generation algorithm)
    - Input: Random Bits  $R$
    - Output:  $(pk, sk) \in \mathcal{K}$
  - $\text{Encaps}_{pk}(1^n, R)$ 
    - Input: security parameter, random bits  $R$
    - Output: Symmetric key  $k \in \{0,1\}^{\ell(n)}$  and a ciphertext  $c$
  - $\text{Decaps}_{sk}(c)$  (Deterministic algorithm)
    - Input: Secret key  $sk \in \mathcal{K}$  and a ciphertext  $c$
    - Output: a symmetric key  $\{0,1\}^{\ell(n)}$  or  $\perp$  (fail)
- **Invariant:**  $\text{Decaps}_{sk}(c)=k$  whenever  $(c,k) = \text{Encaps}_{pk}(1^n, R)$

# KEM CCA-Security ( $\text{KEM}_{A,\Pi}^{\text{cca}}(n)$ )



$$\forall PPT A \exists \mu \text{ (negligible) s.t.}$$

$$\Pr[\text{KEM}_{A,\Pi}^{\text{cca}} = 1] \leq \frac{1}{2} + \mu(n)$$

Random bit  $b$   
 $(pk, sk) = \text{Gen}(\cdot)$



$(c, k_0) = \text{Encaps}_{pk}(\cdot)$   
 $k_1 \leftarrow \{0, 1\}^n$

# CCA-Secure Encryption from CCA-Secure KEM

$$\mathbf{Enc}_{pk}(m; R) = \langle c, \mathbf{Enc}_k^*(m) \rangle$$

Where

- $(c, k) \leftarrow \mathbf{Encaps}_{pk}(\mathbf{1}^n; R)$ ,
- $\mathbf{Enc}_k^*$  is a CCA-Secure symmetric key encryption algorithm, and
- $\mathbf{Encaps}_{pk}$  is a CCA-Secure KEM.

**Theorem 11.14:**  $\mathbf{Enc}_{pk}$  is CCA-Secure public key encryption scheme.

# CCA-Secure KEM in the Random Oracle Model

- Let  $(N, e, d)$  be an RSA key ( $pk = (N, e)$ ,  $sk = (N, d)$ ).

$$\text{Encaps}_{pk}(1^n, R) = (r^e \bmod N, k = H(r))$$

- Remark 1:  $k$  is completely random string unless the adversary can query random oracle  $H$  on input  $r$ .
- Remark 2: If Plain-RSA is hard to invert for a random input then PPT attacker finds  $r$  with negligible probability.

# Week 12 Topic 3: El-Gamal Encryption

# A Quick Remark about Groups

- Let  $\mathbb{G}$  be a group with order  $m = |\mathbb{G}|$  with a binary operation  $\circ$  (over  $\mathbb{G}$ ) and let  $g, h \in \mathbb{G}$  be given and consider sampling  $k \in \mathbb{G}$  uniformly at random then we have

$$\Pr_{k \leftarrow \mathbb{G}}[k = g] = \frac{1}{m}$$

**Question:** What is  $\Pr_{k \leftarrow \mathbb{G}}[k \circ h = g] = \frac{1}{m}$ ?

**Answer:**

$$\Pr_{k \leftarrow \mathbb{G}}[k \circ h = g] = \Pr_{k \leftarrow \mathbb{G}}[k = g \circ h^{-1}] = \frac{1}{m}$$

# A Quick Remark about Groups

**Lemma 11.15:** Let  $\mathbb{G}$  be a group with order  $m = |\mathbb{G}|$  with a binary operation  $\circ$  (over  $G$ ) then for any pair  $g, h \in \mathbb{G}$  we have

$$\Pr_{k \leftarrow \mathbb{G}}[k \circ h = g] = \frac{1}{m}$$

**Remark:** This lemma gives us a way to construct perfectly secret private-key crypto scheme. How?



# El-Gamal Encryption

- Key Generation ( $\text{Gen}(1^n)$ ):
  1. Run  $\mathcal{G}(1^n)$  to obtain a cyclic group  $\mathbb{G}$  of order  $q$  (with  $\|q\| = n$ ) and a generator  $g$  such that  $\langle g \rangle = \mathbb{G}$ .
  2. Choose a random  $x \in \mathbb{Z}_q$  and set  $h = g^x$
  3. Public Key:  $\text{pk} = \langle \mathbb{G}, q, g, h \rangle$
  4. Private Key:  $\text{sk} = \langle \mathbb{G}, q, g, x \rangle$
- $\text{Enc}_{\text{pk}}(m) = \langle g^y, m \cdot h^y \rangle$  for a random  $y \in \mathbb{Z}_q$
- $\text{Dec}_{\text{sk}}(c = (c_1, c_2)) = c_2 c_1^{-x}$

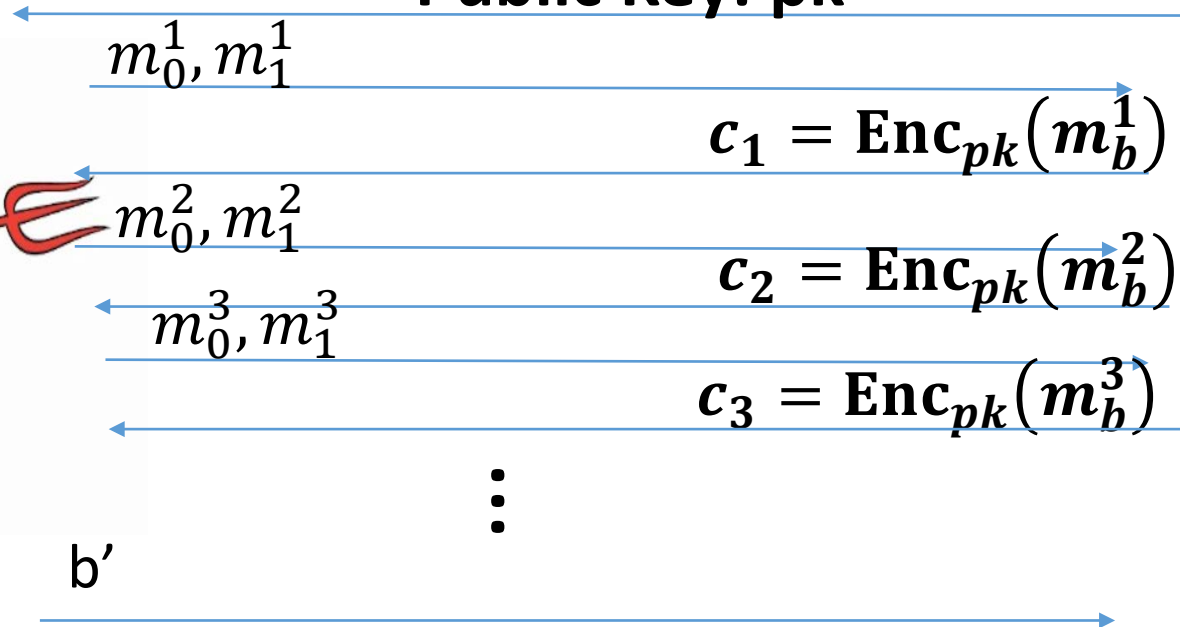
# El-Gamal Encryption

- $\text{Enc}_{\text{pk}}(m) = \langle g^y, m \cdot h^y \rangle$  for a random  $y \in \mathbb{Z}_q$
- $\text{Dec}_{\text{sk}}(c = (c_1, c_2)) = c_2 c_1^{-x}$

$$\begin{aligned}\text{Dec}_{\text{sk}}(g^y, m \cdot h^y) &= m \cdot h^y (g^y)^{-x} \\ &= m \cdot h^y (g^y)^{-x} \\ &= m \cdot (g^x)^y (g^y)^{-x} \\ &= m \cdot g^{xy} g^{-xy} \\ &= m\end{aligned}$$

# CPA-Security ( $\text{PubK}_{A,\Pi}^{\text{LR-cpa}}(n)$ )

Public Key:  $pk$



Random bit  $b$   
 $(pk, sk) = \text{Gen}(\cdot)$



$$\forall PPT A \exists \mu \text{ (negligible) s.t.}$$

$$\Pr[\text{PubK}_{A,\Pi}^{\text{LR-cpa}}(n) = 1] \leq \frac{1}{2} + \mu(n)$$

# El-Gamal Encryption

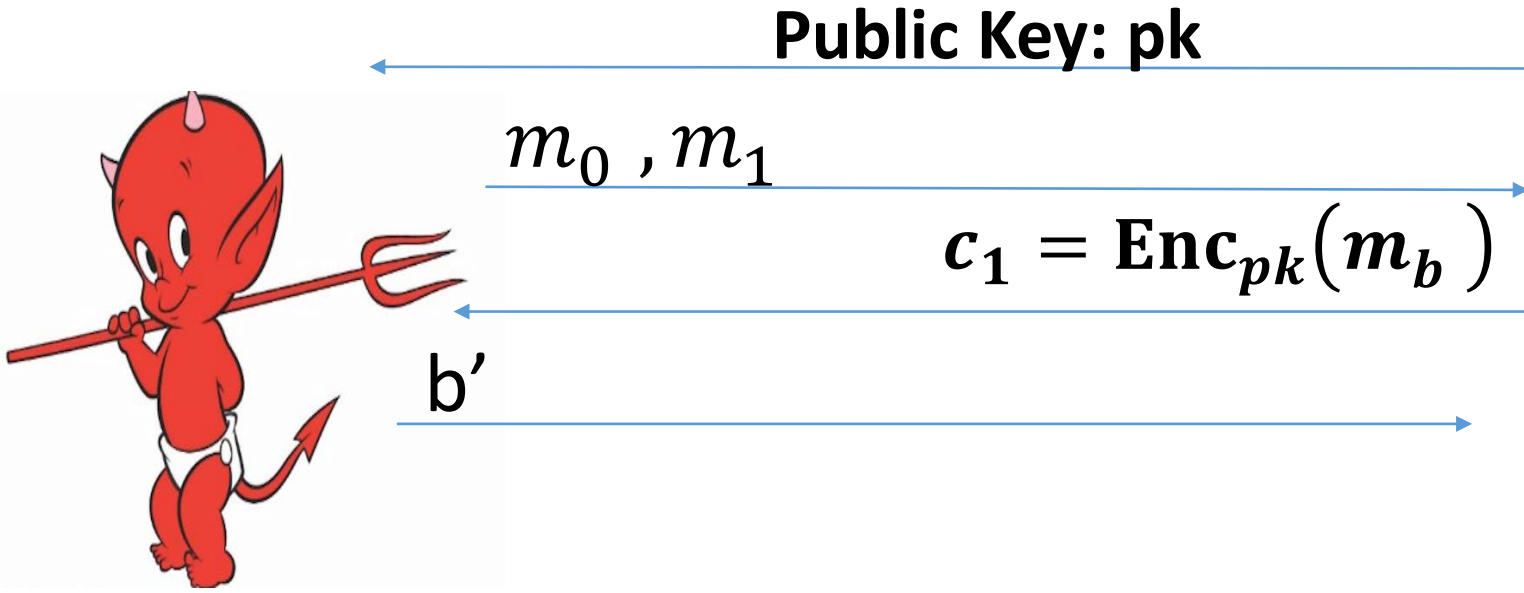
- $\text{Enc}_{\text{pk}}(m) = \langle g^y, m \cdot h^y \rangle$  for a random  $y \in \mathbb{Z}_q$
- $\text{Dec}_{\text{sk}}(c = (c_1, c_2)) = c_2 c_1^{-x}$

**Theorem 11.18:** Let  $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$  be the El-Gamal Encryption scheme (above) then if DDH is hard relative to  $\mathcal{G}$  then  $\Pi$  is CPA-Secure.

**Proof:** Recall that CPA-security and eavesdropping security are equivalent for public key crypto. It suffices to show that for all PPT  $A$  there is a negligible function **negl** such that

$$\Pr[\text{PubK}_{A, \Pi}^{\text{eav}}(n) = 1] \leq \frac{1}{2} + \text{negl}(n)$$

# Eavesdropping Security ( $\text{PubK}_{A,\Pi}^{\text{eav}}(n)$ )



Random bit  $b$   
 $(pk, sk) = \text{Gen}(\cdot)$



$$\forall PPT A \exists \mu \text{ (negligible) s.t.}$$
$$\Pr[\text{PubK}_{A,\Pi}^{\text{eav}}(n) = 1] \leq \frac{1}{2} + \mu(n)$$

# El-Gamal Encryption

**Theorem 11.18:** Let  $\Pi = (Gen, Enc, Dec)$  be the El-Gamal Encryption scheme (above) then if DDH is hard relative to  $\mathcal{G}$  then  $\Pi$  is CPA-Secure.

**Proof:** First introduce an 'encryption scheme'  $\tilde{\Pi}$  in which  $\widetilde{Enc}_{pk}(m) = \langle g^y, m \cdot g^z \rangle$  for random  $y, z \in \mathbb{Z}_q$  (there is actually no way to do decryption, but the experiment  $\text{PubK}_{A, \tilde{\Pi}}^{\text{eav}}(n)$  is still well defined). In fact, (using Lemma 11.15)

$$\begin{aligned} & \Pr[\text{PubK}_{A, \tilde{\Pi}}^{\text{eav}}(n) = 1] \\ &= \frac{1}{2} \Pr[\text{PubK}_{A, \tilde{\Pi}}^{\text{eav}}(n) = 1 | b = 1] + \frac{1}{2} (1 - \Pr[\text{PubK}_{A, \tilde{\Pi}}^{\text{eav}}(n) = 1 | b = 0]) \\ &= \frac{1}{2} + \frac{1}{2} \Pr_{y, z \leftarrow \mathbb{Z}_q} [A(\langle g^y, m \cdot g^z \rangle) = 1] - \frac{1}{2} \Pr_{y, z \leftarrow \mathbb{Z}_q} [A(\langle g^y, g^z \rangle) = 1] \\ &= \frac{1}{2} \end{aligned}$$

# El-Gamal Encryption

**Theorem 11.18:** Let  $\Pi = (Gen, Enc, Dec)$  be the El-Gamal Encryption scheme (above) then if DDH is hard relative to  $\mathcal{G}$  then  $\Pi$  is CPA-Secure.

**Proof:** We just showed that

$$\Pr[\text{PubK}_{A,\tilde{\Pi}}^{\text{eav}}(n) = 1] = \frac{1}{2}$$

Therefore, it suffices to show that

$$|\Pr[\text{PubK}_{A,\Pi}^{\text{eav}}(n) = 1] - \Pr[\text{PubK}_{A,\tilde{\Pi}}^{\text{eav}}(n) = 1]| \leq \mathbf{negl}(n)$$

This, will follow from DDH assumption.

# El-Gamal Encryption

**Theorem 11.18:** Let  $\Pi = (Gen, Enc, Dec)$  be the El-Gamal Encryption scheme (above) then if DDH is hard relative to  $\mathcal{G}$  then  $\Pi$  is CPA-Secure.

**Proof:** We can build  $B(g^x, g^y, Z)$  to break DDH assumption if  $\Pi$  is not CPA-Secure. Simulate eavesdropping attacker  $A$

1. Send attacker public key  $pk = \langle \mathbb{G}, q, g, h = g^x \rangle$
2. Receive  $m_0, m_1$  from  $A$ .
3. Send  $A$  the ciphertext  $\langle g^y, m_b \cdot Z \rangle$ .
4. Output 1 if and only if attacker outputs  $b' = b$ .

$$\begin{aligned} & \left| \Pr[B(g^x, g^y, Z) = 1 \mid Z = g^{xy}] - \Pr[B(g^x, g^y, Z) = 1 \mid Z = g^z] \right| \\ &= \left| \Pr[\text{PubK}_{A, \Pi}^{\text{eav}}(n) = 1] - \Pr[\text{PubK}_{A, \tilde{\Pi}}^{\text{eav}}(n) = 1] \right| \\ &= \left| \Pr[\text{PubK}_{A, \Pi}^{\text{eav}}(n) = 1] - 1/2 \right| \end{aligned}$$



# El-Gamal Encryption

- $\text{Enc}_{\text{pk}}(m) = \langle g^y, m \cdot h^y \rangle$  for a random  $y \in \mathbb{Z}_q$  and  $h = g^x$ ,
- $\text{Dec}_{\text{sk}}(c = (c_1, c_2)) = c_2 c_1^{-x}$

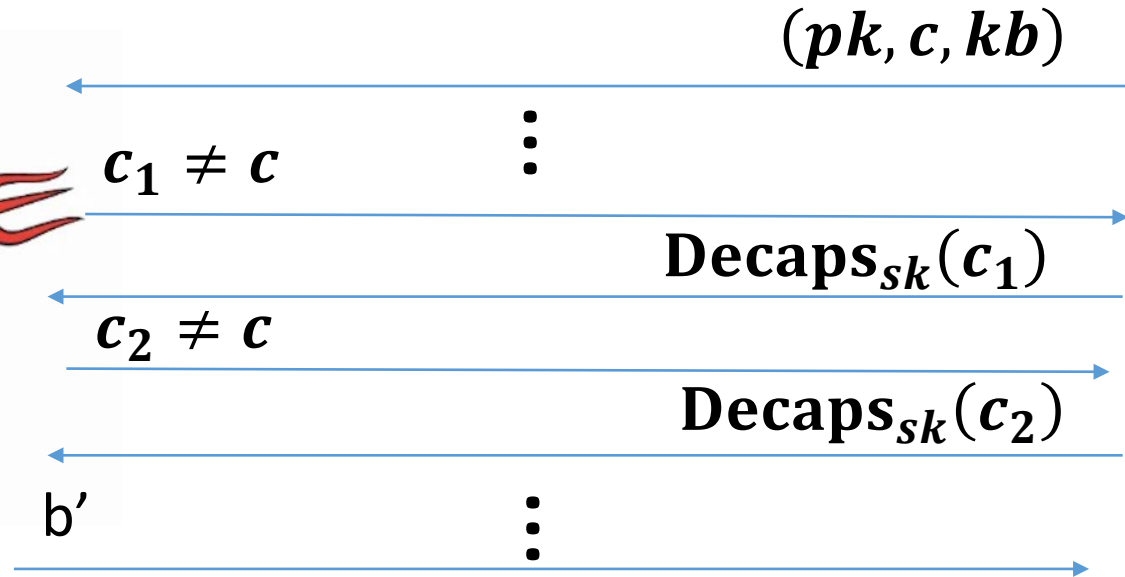
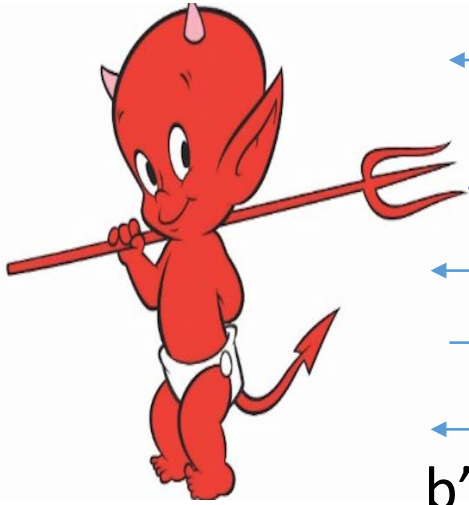
**Fact:** El-Gamal Encryption is malleable.

$$\begin{aligned}c &= \text{Enc}_{\text{pk}}(m) = \langle g^y, m \cdot h^y \rangle \\c' &= \text{Enc}_{\text{pk}}(m) = \langle g^y, 2 \cdot m \cdot h^y \rangle \\ \text{Dec}_{\text{sk}}(c') &= 2 \cdot m \cdot h^y \cdot g^{-xy} = 2m\end{aligned}$$

# Key Encapsulation Mechanism (KEM)

- Three Algorithms
  - $\text{Gen}(1^n, R)$  (Key-generation algorithm)
    - Input: Random Bits  $R$
    - Output:  $(pk, sk) \in \mathcal{K}$
  - $\text{Encaps}_{pk}(1^n, R)$ 
    - Input: security parameter, random bits  $R$
    - Output: Symmetric key  $k \in \{0,1\}^{\ell(n)}$  and a ciphertext  $c$
  - $\text{Decaps}_{sk}(c)$  (Deterministic algorithm)
    - Input: Secret key  $sk \in \mathcal{K}$  and a ciphertext  $c$
    - Output: a symmetric key  $\{0,1\}^{\ell(n)}$  or  $\perp$  (fail)
- **Invariant:**  $\text{Decaps}_{sk}(c)=k$  whenever  $(c,k) = \text{Encaps}_{pk}(1^n, R)$

# KEM CCA-Security ( $\text{KEM}_{A,\Pi}^{\text{cca}}(n)$ )



$$\forall PPT A \exists \mu \text{ (negligible) s.t.}$$

$$\Pr[\text{KEM}_{A,\Pi}^{\text{cca}} = 1] \leq \frac{1}{2} + \mu(n)$$

Random bit  $b$   
 $(pk, sk) = \text{Gen}(\cdot)$

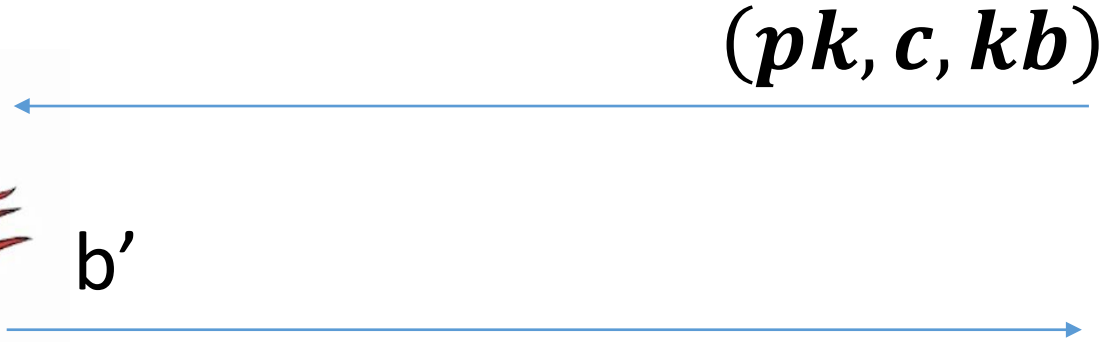
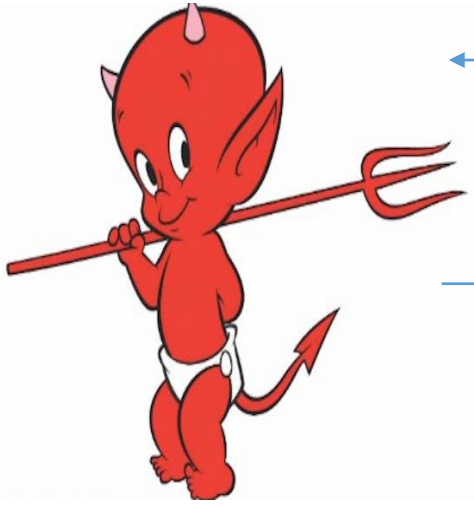


$(c, k_0) = \text{Encaps}_{pk}(\cdot)$   
 $k_1 \leftarrow \{0, 1\}^n$

# Recall: Last Lecture

- CCA-Secure KEM from RSA in Random Oracle Model
- What if we want security proof in the standard model?
- Answer: DDH yields a CPA-Secure KEM in standard model

# KEM CPA-Security ( $\text{KEM}_{A,\Pi}^{\text{cpa}}(n)$ )



$$\forall PPT A \exists \mu \text{ (negligible) s.t.}$$

$$\Pr[\text{KEM}_{A,\Pi}^{\text{cpa}} = 1] \leq \frac{1}{2} + \mu(n)$$

Random bit  $b$   
 $(pk, sk) = \text{Gen}(\cdot)$



$(c, k_0) = \text{Encaps}_{pk}(\cdot)$   
 $k_1 \leftarrow \{0, 1\}^n$

# CCA-Secure Encryption from CPA-Secure KEM

$$\mathbf{Enc}_{pk}(m; R) = \langle c, \mathbf{Enc}_k^*(m) \rangle$$

Where

- $(c, k) \leftarrow \mathbf{Encaps}_{pk}(\mathbf{1}^n; R)$ ,
- $\mathbf{Enc}_k^*$  is a eavesdropping-secure symmetric key encryption algorithm
- $\mathbf{Encaps}_{pk}$  is a CPA-Secure KEM.

**Theorem 11.12:**  $\mathbf{Enc}_{pk}$  is CCA-Secure public key encryption scheme.

# CPA-Secure KEM with El-Gamal

- $\text{Gen}(1^n, R)$  (Key-generation algorithm)
  1. Run  $\mathcal{G}(1^n)$  to obtain a cyclic group  $\mathbb{G}$  of order  $q$  (with  $\|q\| = 2n$ ) and a generator  $g$  such that  $\langle g \rangle = \mathbb{G}$ .
  2. Choose a random  $x \in \mathbb{Z}_q$  and set  $h = g^x$
  3. Public Key:  $\text{pk} = \langle \mathbb{G}, q, g, h \rangle$
  4. Private Key:  $\text{sk} = \langle \mathbb{G}, q, g, x \rangle$
- $\text{Encaps}_{\text{pk}}(1^n, R)$ 
  - Pick random  $y \in \mathbb{Z}_q$
  - Output:  $\langle g^y, k = \text{LeastSigNBits}(h^y) \rangle$
- $\text{Decaps}_{\text{sk}}(c)$  (Deterministic algorithm)
  - Output:  $k = \text{LeastSigNBits}(c^x)$

# CPA-Secure KEM with El-Gamal

- $\text{Gen}(1^n, R)$  (Key-generation algorithm)
  1. Run  $\mathcal{G}(1^n)$  to obtain a cyclic group  $\mathbb{G}$  of order  $q$  (with  $\|\mathbb{G}\| = 2n$ ) and a generator  $g$  such that  $\langle g \rangle = \mathbb{G}$ .
  2. Choose a random  $x \in \mathbb{Z}_q$  and set  $h = g^x$
  3. Public Key:  $\text{pk} = \langle \mathbb{G}, q, g, h \rangle$
  4. Private Key:  $\text{sk} = \langle \mathbb{G}, q, g, x \rangle$
- $\text{Encaps}_{\text{pk}}(1^n, R)$ 
  - Pick random  $y \in \mathbb{Z}_q$
  - Output:  $\langle g^y, k = \text{LeastSigNBits}(h^y) \rangle$
- $\text{Decaps}_{\text{sk}}(c)$  (Deterministic algorithm)
  - Output:  $k = \text{LeastSigNBits}(c^x)$

$$\text{Decaps}_{\text{sk}}(g^y) = \text{LeastSigNBits}(g^{xy}) = \text{LeastSigNBits}(h^y) = k$$



# CPA-Secure KEM with El-Gamal

- $\text{Gen}(1^n, R)$  (Key-generation algorithm)
  1. Run  $\mathcal{G}(1^n)$  to obtain a cyclic group  $\mathbb{G}$  of order  $q$  (with  $\|q\| = 2n$ ) and a generator  $g$  such that  $\langle g \rangle = \mathbb{G}$ .
  2. Choose a random  $x \in \mathbb{Z}_q$  and set  $h = g^x$
  3. Public Key:  $\text{pk} = \langle \mathbb{G}, q, g, h \rangle$
  4. Private Key:  $\text{sk} = \langle \mathbb{G}, q, g, x \rangle$
- $\text{Encaps}_{\text{pk}}(1^n, R)$ 
  - Pick random  $y \in \mathbb{Z}_q$
  - Output:  $\langle g^y, k = \text{LeastSigNBits}(h^y) \rangle$
- $\text{Decaps}_{\text{sk}}(c)$  (Deterministic algorithm)
  - Output:  $k = \text{LeastSigNBits}(c^x)$

**Theorem 11.20:** If DDH is hard relative to  $\mathcal{G}$  then  $(\text{Gen}, \text{Encaps}, \text{Decaps})$  is a CPA-Secure KEM

# CPA-Secure KEM with El-Gamal

- $\text{Gen}(1^n, R)$  (Key-generation algorithm)
  1. Run  $\mathcal{G}(1^n)$  to obtain a cyclic group  $\mathbb{G}$  of order  $q$  (with  $\|q\| = 2n$ ) and a generator  $g$  such that  $\langle g \rangle = \mathbb{G}$ .
  2. Choose a random  $x \in \mathbb{Z}_q$  and set  $h = g^x$
  3. Public Key:  $\text{pk} = \langle \mathbb{G}, q, g, h \rangle$
  4. Private Key:  $\text{sk} = \langle \mathbb{G}, q, g, x \rangle$
- $\text{Encaps}_{\text{pk}}(1^n, R)$ 
  - Pick random  $y \in \mathbb{Z}_q$
  - Output:  $\langle g^y, k = \text{LeastSigNBits}(h^y) \rangle$
- $\text{Decaps}_{\text{sk}}(c)$  (Deterministic algorithm)
  - Output:  $k = \text{LeastSigNBits}(c^x)$

**Remark:** If CDH is hard relative to  $\mathcal{G}$  then  $(\text{Gen}, \text{Encaps}, \text{Decaps})$  and we replace  $\text{LeastSigNBits}$  with a random oracle  $H$  then this is a CPA-Secure KEM

(...also CCA-secure under a slightly stronger assumption called gap-CDH)

# CCA-Secure Variant in Random Oracle Model

- Key Generation ( $\text{Gen}(1^n)$ ):
  1. Run  $\mathcal{G}(1^n)$  to obtain a cyclic group  $\mathbb{G}$  of order  $q$  (with  $\|q\| = n$ ) and a generator  $g$  such that  $\langle g \rangle = \mathbb{G}$ .
  2. Choose a random  $x \in \mathbb{Z}_q$  and set  $h = g^x$
  3. Public Key:  $\text{pk} = \langle \mathbb{G}, q, g, h \rangle$
  4. Private Key:  $\text{sk} = \langle \mathbb{G}, q, g, x \rangle$
- $\text{Enc}_{\text{pk}}(m) = \langle g^y, c', \text{Mac}_{K_M}(c') \rangle$  for a random  $y \in \mathbb{Z}_q$  and  $K_E \parallel K_M = H(h^y)$  and  $c' = \text{Enc}'_{K_E}(m)$
- $\text{Dec}_{\text{sk}}(\langle c, c', t \rangle)$ 
  1.  $K_E \parallel K_M = H(c^x)$
  2. If  $\text{Vrfy}_{K_M}(c', t) \neq 1$  or  $c \notin \mathbb{G}$  output  $\perp$ ; otherwise output  $\text{Dec}'_{K_E}(c', t)$

# CCA-Secure Variant in Random Oracle Model

**Theorem:** If  $\text{Enc}'_{K_E}$  is CPA-secure,  $\text{Mac}_{K_M}$  is a strong MAC and a problem called gap-CDH is hard then this a CCA-secure public key encryption scheme in the random oracle model.

- $\text{Enc}_{\text{pk}}(m) = \langle g^y, c', \text{Mac}_{K_M}(c') \rangle$  for a random  $y \in \mathbb{Z}_q$  and  $K_E \parallel K_M = H(h^y)$  and  $c' = \text{Enc}'_{K_E}(m)$
- $\text{Dec}_{\text{sk}}(\langle c, c', t \rangle)$ 
  1.  $K_E \parallel K_M = H(c^x)$
  2. If  $\text{Vrfy}_{K_M}(c', t) \neq 1$  or  $c \notin \mathbb{G}$  output  $\perp$ ; otherwise output  $\text{Dec}'_{K_E}(c', t)$

# CCA-Secure Variant in Random Oracle Model

**Remark:** The CCA-Secure variant is used in practice in the ISO/IEC 18033-2 standard for public-key encryption.

- Diffie-Hellman Integrated Encryption Scheme (DHIES)
- Elliptic Curve Integrated Encryption Scheme (ECIES)
- $\text{Enc}_{\text{pk}}(m) = \langle g^y, c', \text{Mac}_{K_M}(c') \rangle$  for a random  $y \in \mathbb{Z}_q$  and  $K_E \parallel K_M = H(h^y)$  and  $c' = \text{Enc}'_{K_E}(m)$
- $\text{Dec}_{\text{sk}}(\langle c, c', t \rangle)$ 
  1.  $K_E \parallel K_M = H(c^x)$
  2. If  $\text{Vrfy}_{K_M}(c', t) \neq 1$  or  $c \notin \mathbb{G}$  output  $\perp$ ; otherwise output  $\text{Dec}'_{K_E}(c', t)$