# CSE107: Intro to Modern Cryptography

https://cseweb.ucsd.edu/classes/sp22/cse107-a/

Emmanuel Thomé

May 10, 2022

# Lecture 10c

## A few important points about DLog and RSA

# RSA: what to remember

The RSA function $f(x) = x^e \mod N$ is a trapdoor one way permutation:

- Easy forward: given $N, e, x$ it is easy to compute $f(x)$
- Easy back with trapdoor: Given $N, d$ and $y = f(x)$ it is easy to compute $x = f^{-1}(y) = y^d \mod N$
- Hard back without trapdoor: Given $N, e$ and $y = f(x)$ it is hard to compute $x = f^{-1}(y)$

# RSA key sizes

Factoring is hard. The largest RSA key factorization that is publicly known is 829 bits.

Conventional wisdom is that factoring a 1024-bit RSA modulus takes less than $2^{80}$ time, and is well within range of state-level adversaries.

A 2048-bit key is the bare minimum of security recommendations worldwide, but it only provides an estimated 112 bits of security.

In order to have 128 bits of security, a 3072-bit key is necessary. This is large, and imposes a constraint on some devices and protocols.

Caveat: these large-bit-size extrapolations are not based on very firm theoretical ground.

# Exponentiation: what to remember

In a group $G$ of order $\#G$ (example: in $\mathbb{Z}_p^*$, which has order $p-1$), every element has an order, which is a divisor of $\#G$.

- Order of $G$: number of elements in $G$.
- Order of an element $a \in G$: smallest integer $\ell > 0$ such that $a^\ell = \mathbf{id}$.

Let $a \in G$ of order $\ell$. The element $a^k \in G$ can be computed for any $k \in \mathbb{Z}$. Furthermore, since $a$ has order $\ell$:

$$a^k = a^{k+\ell} = a^{k+2\ell} = \cdots$$

Therefore, we may just as well say that we compute $a^k$ for $k \in \mathbb{Z}_{\#G}$, or even for $k \in \mathbb{Z}_\ell$ (which representative we take modulo $\ell$ does not matter).

# DLog: what to remember

In any group that we intend to use in cryptography, the exponentiation problem is easy.

- Square-and-multiply computes $a^k$ from $a \in G$ and $k \in \mathbb{Z}_{\#G}$ in $O(\log k)$ group law operations in $G$.
- For example, in $\mathbb{Z}_p^*$, which has order $p-1$, a group law operation costs $O((\log p)^2)$, and thus exponentiation costs $O((\log p)^3)$ (assuming $k$ and $p-1$ have the same size).

# DLog: what to remember

In contrast, the discrete logarithm problem (DLP) is hard.

- To go from $a^k$ back to $k$, the stupid algorithm works, but it takes forever ($O(\#G)$ operations).
- There are mildly faster algorithms that work in any group (we did not discuss them), but they take $O(\sqrt{\#G})$, which is still exponential in $\log \#G$. On elliptic curves, this is the best we can do.
- In groups like $\mathbb{Z}_p^*$, there are advanced ways to compute discrete logarithms, with sub-exponential complexity. Very roughly, it is

$$e^{1.92(\ln p)^{1/3}(\ln \ln p)^{2/3}}.$$

DLog remains a hard problem nevertheless. Cryptanalysis results and key length recommendations are in line with RSA.

# CSE107: Intro to Modern Cryptography

https://cseweb.ucsd.edu/classes/sp22/cse107-a/

Emmanuel Thomé

May 10, 2022

# Lecture 12

## Hybrid Encryption and KEMs

Building PKE schemes with hybrid encryption

We need KEMs

# Plan

Building PKE schemes with hybrid encryption

We need KEMs

# How do we achieve security

The goals are set. How do we achieve IND-X (e.g., IND-CCA) security?

- We have some building blocks (Diffie-Hellman key exchange, RSA), but the translation to PKE is not immediate.
- We need to take many precautions in the design, because security is not always an easy thing to achieve!
- We want to leverage the existing (secure) building blocks to our advantage.

Main idea:

- Use public-key encryption to establish a (shared, secret) key $K$.
- Use $K$ to encrypt the message with a symmetrickey encryption scheme.

# Hybrid encryption

The task of building an asymmetric encryption scheme $\mathcal{AE}$ is simplified by hybrid encryption. The ingredients are

- A key encapsulation mechanism (KEM) $\mathcal{KE}$
- a symmetric encryption scheme $\mathcal{SE}$.

To encrypt message $M$ under encryption key $ek$:

- Run the key encapsulation mechanism on input $ek$ to obtain a symmetric key $K$ and a ciphertext $C_a$ encrypting it
- Encrypt $M$ with $K$ using the symmetric encryption scheme to get a ciphertext $C_s$
- Return $(C_a, C_s)$ as the ciphertext

Benefits: Modularity of design and analysis, speed.

# Syntax of a Key Encapsulation Mechanism (KEM)

## Definition (Key Encapsulation Mechanism, a.k.a. KEM)

A KEM $\mathcal{KE} = (\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$ is a triple of algorithms. Associated to it is an integer $k$ called the key length. The algorithms operate as follows:

- $(ek, dk) \xleftarrow{\$} \mathcal{KK}$ — generate an encryption key $ek$ and matching decryption key $dk$
- $(K, C_a) \xleftarrow{\$} \mathcal{EK}_{ek}$ — generate a key $K \in \{0,1\}^k$ together with a ciphertext $C_a$ encrypting $K$. Algorithm $\mathcal{EK}$ may be randomized.
- $K' \leftarrow \mathcal{DK}_{dk}(C_a)$ — decrypt ciphertext $C_a$ under decryption key $dk$ to get an output $K' \in \{0,1\}^* \cup \{\perp\}$.

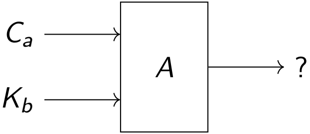
The correct decryption requirement is that, for all $(ek, dk)$ that may be output by $\mathcal{KK}$, we have $K' = K$ with probability 1 when $(K, C_a) \xleftarrow{\$} \mathcal{EK}_{ek}$ and $K' \leftarrow \mathcal{DK}_{dk}(C_a)$.

# KEM Security

Let $\mathcal{KE} = (\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$ be a KEM with key length $k$. Security requires that if we let

$$(K_1, C_a) \xleftarrow{\$} \mathcal{EK}_{ek}$$

then $K_1$ should look "random". Somewhat more precisely, if we also generate $K_0 \xleftarrow{\$} \{0,1\}^k$ ; $b \xleftarrow{\$} \{0,1\}$ then




The adversary $A$ has a hard time figuring out $b$

As we did for symmetric and public-key encryption schemes, we can define security games for KEMs.

# KEM IND-CPA security games

Let $\mathcal{KE} = (\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$ be a KEM with key length $k$.

**Game Left$_{\mathcal{KE}}$**

**procedure Initialize**
$(ek, dk) \xleftarrow{\$} \mathcal{KK}$
return $ek$

**procedure Enc**
$K_0 \xleftarrow{\$} \{0,1\}^k$ ; $(K_1, C_a) \xleftarrow{\$} \mathcal{EK}_{ek}$
return $(K_0, C_a)$

**Game Right$_{\mathcal{KE}}$**

**procedure Initialize**
$(ek, dk) \xleftarrow{\$} \mathcal{KK}$
return $ek$

**procedure Enc**
$K_0 \xleftarrow{\$} \{0,1\}^k$; $(K_1, C_a) \xleftarrow{\$} \mathcal{EK}_{ek}$
return $(K_1, C_a)$

**Definition (ind-cpa advantage $\mathbf{Adv}^{\text{ind-cpa}}$ for KEMs)**

The (ind-cpa) advantage of an adversary $A$ is

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

# KEM IND-CCA security games

Let $\mathcal{KE} = (\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$ be a KEM with key length $k$.

**Game $\mathrm{Left}_{\mathcal{KE}}$**

**procedure Initialize**
$(ek, dk) \xleftarrow{\$} \mathcal{KK}$ ; $S \leftarrow \emptyset$ ; return $ek$

**procedure Enc**
$K_0 \xleftarrow{\$} \{0,1\}^k$ ; $(K_1, C_a) \xleftarrow{\$} \mathcal{EK}_{ek}$
$S \leftarrow S \cup \{C_a\}$ ; return $(K_0, C_a)$

**procedure Dec**$(C_a)$
if $C_a \in S$ then return $\perp$
else $K \leftarrow \mathcal{DK}_{dk}(C_a)$ ; return $K$

**Game $\mathrm{Right}_{\mathcal{KE}}$**

**procedure Initialize**
$(ek, dk) \xleftarrow{\$} \mathcal{KK}$ ; $S \leftarrow \emptyset$ ; return $ek$

**procedure Enc**
$K_0 \xleftarrow{\$} \{0,1\}^k$ ; $(K_1, C_a) \xleftarrow{\$} \mathcal{EK}_{ek}$
$S \leftarrow S \cup \{C_a\}$ ; return $(K_1, C_a)$

**procedure Dec**$(C_a)$
if $C_a \in S$ then return $\perp$
else $K \leftarrow \mathcal{DK}_{dk}(C_a)$ ; return $K$

**Definition (ind-cca advantage $\mathbf{Adv}^{\text{ind-cca}}$ for KEMs)**

The (ind-cca) advantage of an adversary $A$ is

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

# Definitional chart

| | IND-CPA | IND-CCA |
|---|---|---|
| PKE | | |
| KEM | | |
| SE | | |

- Three types of schemes/syntax : Public-key encryption, key encapsulation mechanism, symmetric encryption
- For each, two definitions of security: IND-CPA, IND-CCA

For all three types of schemes/syntax: IND-CCA implies IND-CPA

# Hybrid encryption

Given:
- A KEM $\mathcal{KE} = (\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$ with key length $k$
- A symmetric encryption scheme $\mathcal{SE} = (\mathcal{KS}, \mathcal{ES}, \mathcal{DS})$ for which $\mathcal{KS}$ returns random $k$-bit keys.

Hybrid encryption associates to the above a PKE scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$:

| **Alg** $\mathcal{K}$ | **Alg** $\mathcal{E}_{ek}(M)$ | **Alg** $\mathcal{D}_{dk}((C_a, C_s))$ |
|---|---|---|
| $(ek, dk) \xleftarrow{\$} \mathcal{KK}$ | $(K, C_a) \xleftarrow{\$} \mathcal{EK}_{ek}$ | $K \leftarrow \mathcal{DK}_{dk}(C_a)$ |
| return $(ek, dk)$ | $C_s \xleftarrow{\$} \mathcal{ES}_K(M)$ | $M \leftarrow \mathcal{DS}_K(C_s)$ |
| | return $(C_a, C_s)$ | return $M$ |

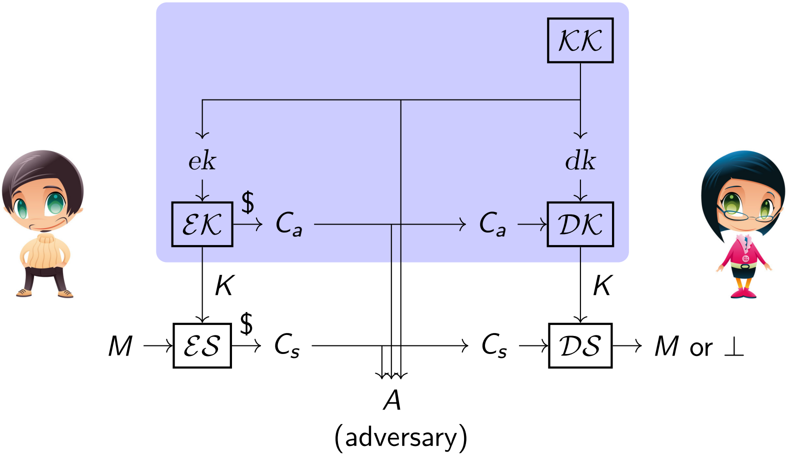
Above, it is understood that if any input to an algorithm is $\perp$, then so is the output.

In layman terms:

- Use the KEM to securely communicate some random encryption key to the receiving party.
- Use the symmetric encryption scheme with the freshly generated key to encrypt the bulk of the message.

# KEMs, graphically

Bob wants to send a message to Alice.



# Hybrid encryption works

If the KEM and symmetric encryption scheme are both IND-X, then so is the PKE scheme constructed by hybrid encryption.

## Theorem

*Let $\mathcal{KE} = (\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$ be a KEM with key length $k$. Let $\mathcal{SE} = (\mathcal{KS}, \mathcal{ES}, \mathcal{DS})$ be a symmetric encryption scheme for which $\mathcal{KS}$ returns random $k$-bit keys. Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the corresponding PKE scheme built via hybrid encryption. Let $\mathrm{X} \in \{\mathrm{cpa}, \mathrm{cca}\}$. Let $A$ be an adversary making $q_e$* ***LR*** *queries. Then there are adversaries $B_a, B_s$ such that*

$$\mathbf{Adv}^{\mathrm{ind\text{-}X}}_{\mathcal{AE}}(A) \leq 2 \cdot \mathbf{Adv}^{\mathrm{ind\text{-}X}}_{\mathcal{KE}}(B_a) + q_e \cdot \mathbf{Adv}^{\mathrm{ind\text{-}X}}_{\mathcal{SE}}(B_s) \; .$$

*The number of* **Enc** *queries of $B_a$ is $q_e$. The number of* ***LR*** *queries of $B_s$ is 1. In the $\mathrm{X} = \mathrm{cca}$ case, $B_a, B_s$ each make the same number of* **Dec** *queries as $A$. The running times of $B_a, B_s$ are about the same as that of $A$.*

# Benefits of hybrid encryption

**Modular design:** Many choices of components, KEMs are simpler than PKE schemes.

**Assurance via proof** as per above Theorem saying hybrid encryption works.

**Speed:** The block ciphers and hash functions used in symmetric cryptography are much faster (factors of 100x to 10,000x depending on platforms) than the operations on numbers used for asymmetric cryptography.

So performance is improved by limiting the number-theoretic operations as in hybrid encryption.

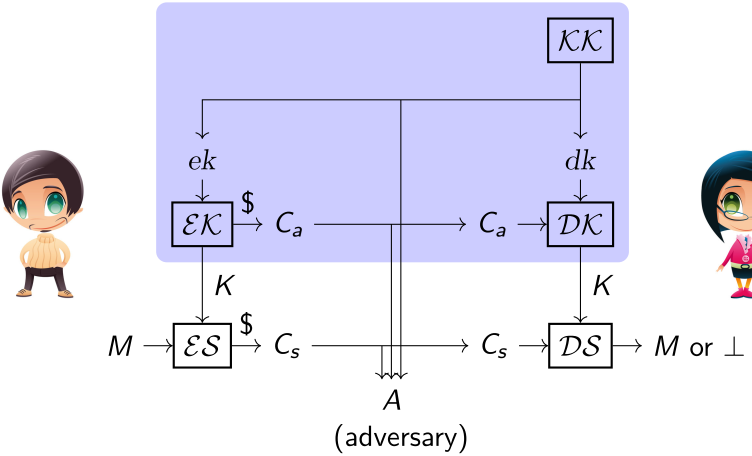
# Plan

Building PKE schemes with hybrid encryption

We need KEMs

# KEMs, graphically

Bob wants to send a message to Alice.



# Where we are

We know how to achieve IND-X-secure PKE given

- An IND-X-secure KEM, and
- An IND-X-secure symmetric encryption scheme

We have plenty of symmetric encryption schemes:

- For the IND-CPA case: AES-CTR\$, AES-CBC\$, ...
- For the IND-CCA case: Encrypt-then-Mac, OCB, GCM, ...

But simpler, deterministic choices are possible too, since security is only required against adversaries $B_s$ making 1 **LR** query. (see Theorem)

We need KEMs.

We will build KEMs using number theory, considering in turn using the DL problem and using RSA.

# Plan

We need KEMs

- Hashing in KEMs, and the Random Oracle Model
- KEMs from DL / KEMs from CDH
- KEMs and PKEs from RSA

# Hashing in KEMs

Our KEMs may use (public, keyless) functions $\mathbf{H}_i$: $\{0,1\}^* \rightarrow \{0,1\}^{\ell_i}$, for $1 \leq i \leq n$.

The number $n$ of them, and their output lengths, depend on the scheme. Usually $n = 1$ or $n = 2$.

In practice (implementation), these are built from cryptographic hash functions as discussed next.

Proofs of security for the KEMs use the Random Oracle Model (ROM) in which $\mathbf{H}_1, \ldots, \mathbf{H}_n$ are modeled as independent random functions.

$\mathbf{H}_1, \ldots, \mathbf{H}_n$ are formalized as game procedures to which scheme algorithms, *as well as the adversary*, have oracle access, and are thus called Random Oracles (ROs).

# Practical choices for the $\mathbf{H}_i$

We seek suitable functions $\mathbf{H}_i$: $\{0,1\}^* \to \{0,1\}^{\ell_i}$, for $1 \le i \le n$.

$\mathrm{SHAKE256}$ is an XOF (eXtendable Output length Function) that takes an input indicating the number of output bits returned.

So we could set $\mathbf{H}_i(x) = \mathrm{SHAKE256}(\langle i \rangle \| x, \ell_i)$ where $\langle i \rangle$ is a 1-byte encoding of $i$ and we assume $n < 2^8$.

We could also set $\mathbf{H}_i(x)$ to the first $\ell_i$ bits of the sequence

$$\mathsf{SHA256}(\langle 0 \rangle \| \langle i \rangle \| x) \,\|\, \mathsf{SHA256}(\langle 1 \rangle \| \langle i \rangle \| x) \,\|\, \cdots \,\|\, \mathsf{SHA256}(\langle 2^8 - 1 \rangle \| \langle i \rangle \| x)$$

This assumes $\ell_i \le 2^8 \cdot 256$.

Heuristically, we desire that $\mathbf{H}_1, \ldots, \mathbf{H}_n$ "behave like independent random functions." But there is no corresponding formal definition.

# Plan

We need KEMs

- Hashing in KEMs, and the Random Oracle Model
- KEMs from DL / KEMs from CDH
- KEMs and PKEs from RSA

# Syntax of a Key Encapsulation Mechanism (KEM)

## Definition (Key Encapsulation Mechanism, a.k.a. KEM)

A KEM $\mathcal{KE} = (\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$ is a triple of algorithms. Associated to it is an integer $k$ called the key length. The algorithms operate as follows:

- $(ek, dk) \xleftarrow{\$} \mathcal{KK}$ — generate an encryption key $ek$ and matching decryption key $dk$
- $(K, C_a) \xleftarrow{\$} \mathcal{EK}_{ek}$ — generate a key $K \in \{0,1\}^k$ together with a ciphertext $C_a$ encrypting $K$. Algorithm $\mathcal{EK}$ may be randomized.
- $K' \leftarrow \mathcal{DK}_{dk}(C_a)$ — decrypt ciphertext $C_a$ under decryption key $dk$ to get an output $K' \in \{0,1\}^* \cup \{\perp\}$.

The correct decryption requirement is that, for all $(ek, dk)$ that may be output by $\mathcal{KK}$, we have $K' = K$ with probability 1 when $(K, C_a) \xleftarrow{\$} \mathcal{EK}_{ek}$ and $K' \leftarrow \mathcal{DK}_{dk}(C_a)$.

# KEMs from DL?

Let $G = \langle g \rangle$ be a cyclic group of order $m$ in which the DL problem is hard. Can we design a KEM $\mathcal{KE} = (\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$ whose IND-CPA security reduces to DL?

**How about:** Let the receiver's encryption key be $g$. Let $\mathcal{EK}_g$ pick $x \xleftarrow{\$} \mathbb{Z}_m$ and return $(x, X)$ where $X = g^x$.

Then obtaining $x$ from $X$ requires solving DL, and would be hard for an adversary.

So are we done?

# KEMs from DL?

Let $G = \langle g \rangle$ be a cyclic group of order $m$ in which the DL problem is hard. Can we design a KEM $\mathcal{KE} = (\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$ whose IND-CPA security reduces to DL?

**How about:** Let the receiver's encryption key be $g$. Let $\mathcal{EK}_g$ pick $x \xleftarrow{\$} \mathbb{Z}_m$ and return $(x, X)$ where $X = g^x$.

Then obtaining $x$ from $X$ requires solving DL, and would be hard for an adversary.

So are we done?

**No.** The legitimate receiver has no way to decrypt $X$, to obtain $x$, short of computing DL.

A sign that something is amiss is that, in the above scheme, the receiver has no decryption key.

# Recall DHSecret Key Exchange

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

| **Alice** | | **Bob** |
|---|---|---|
| $x \xleftarrow{\$} \mathbb{Z}_m$; $X \leftarrow g^x \bmod p$ | | |
| | $\xrightarrow{X}$ | |
| | | $y \xleftarrow{\$} \mathbb{Z}_m$; $Y \leftarrow g^y$ |
| | $\xleftarrow{Y}$ | |
| $K_A \leftarrow Y^x$ | | $K_B \leftarrow X^y$ |

- $Y^x = (g^y)^x = g^{xy} = (g^x)^y = X^y$, so $K_A = K_B$
- Adversary is faced with the CDH problem, which needs to be assumed hard for security. This is a stronger requirement than hardness of DL.

# From key exchange to PKE

We can turn $\mathrm{DH}$key exchange into a PKE scheme via

- Alice has encryption key $X = g^x$ and decryption key $x \xleftarrow{\$} \mathbb{Z}_{p-1}$
- If Bob wants to encrypt message $M$ for Alice, he
  - Picks $y \xleftarrow{\$} \mathbb{Z}_{p-1}$ and sends $Y = g^y$ to Alice
  - Computes $Z = (g^x)^y = g^{xy}$, hashes it to get a key $K$, encrypts $M$ symmetrically under $K$ to get a ciphertext $C_s$, and sends $C_s$ to Alice.
- Alice can recompute $Z = Y^x = g^{xy}$ using her decryption key $x$. Then she can recompute $K$ and decrypt $C_s$ under it to get $M$.

The adversary is faced with either solving $\mathrm{CDH}$ or breaking the symmetric encryption scheme.

# The DHIES scheme

Let $G = \langle g \rangle$ be a cyclic group of order $m$ and $\mathbf{H}\colon G \to \{0,1\}^n$ a (public) hash function. The DHIES PKE scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined for messages $M \in \{0,1\}^n$ via

| **Alg** $\mathcal{K}$ | **Alg** $\mathcal{E}_X(M)$ | **Alg** $\mathcal{D}_x(Y, W)$ |
|---|---|---|
| $x \xleftarrow{\$} \mathbb{Z}_m$ | $y \xleftarrow{\$} \mathbb{Z}_m$; $Y \leftarrow g^y$ | $K \leftarrow Y^x$ |
| $X \leftarrow g^x$ | $K \leftarrow X^y$ | $M \leftarrow \mathbf{H}(K) \oplus W$ |
| return $(X, x)$ | $W \leftarrow \mathbf{H}(K) \oplus M$ | return $M$ |
| | return $(Y, W)$ | |

Correct decryption is assured because $K = X^y = g^{xy} = Y^x$

**Note:** This is a simplified version of the actual scheme.

# DHIES as Hybrid Encryption

DHIES is built along the lines of the Hybrid encryption mechanism.

Pattern matching exercise:

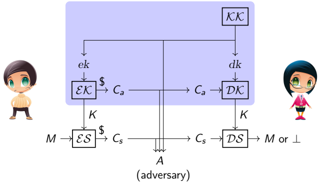
- Can you write down the underlying KEM and its algorithms $(\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$, based on the description of the resulting PKE?
- What is the underlying symmetric encryption scheme?

# DHIES as Hybrid Encryption

Can you write down the KEM inside DHIES and its algorithms $(\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$, based on the description of DHIES as the resulting PKE?

Bob talks to Alice.

- Who runs $\mathcal{KK}$?

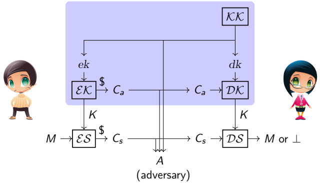


# DHIES as Hybrid Encryption

Can you write down the KEM inside DHIES and its algorithms ($\mathcal{KK}, \mathcal{EK}, \mathcal{DK}$), based on the description of DHIES as the resulting PKE?

Bob talks to Alice.

- Who runs $\mathcal{KK}$? Alice.
  Alice keeps $dk$. Bob gets $ek$.
- Who runs $\mathcal{EK}$?

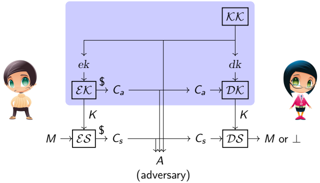


# DHIES as Hybrid Encryption

Can you write down the KEM inside DHIES and its algorithms $(\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$, based on the description of DHIES as the resulting PKE?

Bob talks to Alice.

- Who runs $\mathcal{KK}$? Alice.
  Alice keeps $dk$. Bob gets $ek$.
- Who runs $\mathcal{EK}$? Bob.
  $\mathcal{EK}$ uses $ek$.
- $ek$ must be $X$; $dk$ must be $x$.

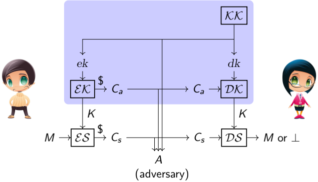


# DHIES as Hybrid Encryption

Can you write down the KEM inside DHIES and its algorithms $(\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$, based on the description of DHIES as the resulting PKE?

Bob talks to Alice.

- Who runs $\mathcal{KK}$? Alice.
  Alice keeps $dk$. Bob gets $ek$.
- Who runs $\mathcal{EK}$? Bob.
  $\mathcal{EK}$ uses $ek$.
- $ek$ must be $X$; $dk$ must be $x$.




The output of $\mathcal{EK}$ consists of:

- $K$ = what Bob will use to encrypt the plaintext.
  Notations aren't too bad, it's $K = X^y$.
- $C_a$ = the encapsulation of $K$ that Alice should decrypt prior to attempting the decryption of $C_s$. The encapsulation is... $Y$, because Alice can recover $K = X^y = g^{xy}$ from just $Y$ and $sk$.

# Security of DHIES

The DHIES scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ associated to cyclic group $G = \langle g \rangle$ and (public) hash function **H** can be proven IND-CPA assuming

- CDH is hard in $G$, and
- **H** is a "random oracle," meaning a "perfect" hash function.

Our simplified version does not make it easy to prove IND-CCA security, and the more complete version of the protocol is designed to make that possible.

# ECIES

ECIES is DHIES with the group being an elliptic curve group.

ECIES features:

| Operation | Cost |
| --- | --- |
| encryption | 2 256-bit exp |
| decryption | 1 256-bit exp |
| ciphertext expansion | 256 bits |

ciphertext expansion = (length of ciphertext) - (length of plaintext)

# Plan

We need KEMs

- Hashing in KEMs, and the Random Oracle Model
- KEMs from DL / KEMs from CDH
- KEMs and PKEs from RSA

# Plain-RSA PKE scheme

Let $\mathcal{K}_{\rm rsa}$ be an RSA generator.
The plain RSA PKE scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined via:

| | | |
|---|---|---|
| **Alg** $\mathcal{K}$ | **Alg** $\mathcal{E}_{(N,e)}(M)$ | **Alg** $\mathcal{D}_{(N,d)}(C)$ |
| $(N, p, q, e, d) \xleftarrow{\$} \mathcal{K}_{\rm rsa}$ | $C \leftarrow M^e \bmod N$ | $M \leftarrow C^d \bmod N$ |
| Return $((N, e), (N, d))$ | return $C$ | return $M$ |

Above, $(N, e)$ is the encryption key and $(N, d)$ is the decryption key.

**Decryption correctness:** The "easy-backwards with trapdoor" property implies that for all $M \in \mathbb{Z}_N^*$ we have $\mathcal{D}_{dk}(\mathcal{E}_{ek}(M)) = M$.

**Note:** The message space is $\mathbb{Z}_N^*$. Messages are assumed to be all encoded as strings of the same length, for example length 4 if $N = 15$.

# Security of the Plain-RSA PKE scheme

Let $\mathcal{K}_{\rm rsa}$ be an RSA generator.
The plain RSA PKE scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined via:

| **Alg** $\mathcal{K}$ | **Alg** $\mathcal{E}_{(N,e)}(M)$ | **Alg** $\mathcal{D}_{(N,d)}(C)$ |
|---|---|---|
| $(N, p, q, e, d) \xleftarrow{\$} \mathcal{K}_{\rm rsa}$ | $C \leftarrow M^e \bmod N$ | $M \leftarrow C^d \bmod N$ |
| Return $((N, e), (N, d))$ | return $C$ | return $M$ |

Getting $d$ from $(N, e)$ involves factoring $N$.

- But $\mathcal{E}$ is deterministic so. . .

# Security of the Plain-RSA PKE scheme

Let $\mathcal{K}_{\rm rsa}$ be an RSA generator.
The plain RSA PKE scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined via:

| **Alg** $\mathcal{K}$ | **Alg** $\mathcal{E}_{(N,e)}(M)$ | **Alg** $\mathcal{D}_{(N,d)}(C)$ |
|---|---|---|
| $(N, p, q, e, d) \stackrel{\$}{\leftarrow} \mathcal{K}_{\rm rsa}$ | $C \leftarrow M^e \bmod N$ | $M \leftarrow C^d \bmod N$ |
| Return $((N, e), (N, d))$ | return $C$ | return $M$ |

Getting $d$ from $(N, e)$ involves factoring $N$.

- But $\mathcal{E}$ is deterministic so... we can detect repeats and the scheme is not IND-CPA secure.
- Plain RSA as a PKE has many other flaws, such as malleability.

# The SRSA scheme

The SRSA PKE scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ associated to RSA generator $\mathcal{K}_{\mathrm{rsa}}$ and (public) hash function $\mathbf{H}$: $\{0,1\}^* \to \{0,1\}^n$ encrypts $n$-bit messages via:

| **Alg** $\mathcal{K}$ | **Alg** $\mathcal{E}_{N,e}(M)$ | **Alg** $\mathcal{D}_{N,d}(C_a, C_s)$ |
|---|---|---|
| $(N, p, q, e, d) \xleftarrow{\$} \mathcal{K}_{\mathrm{rsa}}$ | $x \xleftarrow{\$} \mathbb{Z}_N^*$ | $x \leftarrow C_a^d \bmod N$ |
| $ek \leftarrow (N, e)$ | $K \leftarrow \mathbf{H}(x)$ | $K \leftarrow \mathbf{H}(x)$ |
| $dk \leftarrow (N, d)$ | $C_a \leftarrow x^e \bmod N$ | $M \leftarrow C_s \oplus K$ |
| return $(ek, dk)$ | $C_s \leftarrow K \oplus M$ | return $M$ |
| | return $(C_a, C_s)$ | |

# SRSA as Hybrid Encryption

SRSA is built along the lines of the Hybrid encryption mechanism.

Pattern matching exercise:

- Can you write down the underlying KEM and its algorithms $(\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$, based on the description of the resulting PKE?
- What is the underlying symmetric encryption scheme?

# Security of SRSA

The SRSA PKE scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ associated to RSA generator $\mathcal{K}_{\rm rsa}$ and (public) hash function $\mathbf{H}$: $\{0,1\}^* \to \{0,1\}^n$ can be proven IND-CPA assuming

- $\mathcal{K}_{\rm rsa}$ is one-way
- **H** is a "random oracle," meaning a "perfect" hash function.

# SRSA features

In order to have 128-bit security, RSA keys must be as large as 3072 bits.

SRSA features:

| **Operation** | **Cost** |
|---|---|
| encryption | 1 small exponentiation modulo a 3072-bit $N$ |
| decryption | 1 large exponentiation modulo a 3072-bit $N$ |
| ciphertext expansion | 3072 bits |

ciphertext expansion = (length of ciphertext) - (length of plaintext)

# PKE summary

| **Scheme** | **IND-CPA**? |
|---|---|
| DHIES | Yes |
| Plain RSA | No |
| SRSA | Yes |
| RSA OAEP | Yes |

# KEMs summary

- KEMs are used inside Hybrid Encryption.
- We have proper security notions for KEMs, for symmetric encryption schemes, and that leads to proper security notions for the resulting PKE.
- In several cases, writing down the KEM part of a PKE seems a bit artificial, given that a DH Key exchange or the RSA functions can do a lot more than a KEM.
- Some other cryptographic primitives, however (esp. in the post-quantum setting) only define KEMs, and that is fine.

# Exercise

Let $m, k, \ell$ be integers such that $2 \le m < k$ and $k \ge 2048$ and $\ell = k - m - 1$ and $\ell$ is even. Let $\mathcal{K}_{\mathrm{rsa}}$ be a RSA generator with associated security parameter $k$. Consider the key-generation and encryption algorithms below, where $M \in \{0,1\}^m$:

| **Alg** $\mathcal{K}$ | **Alg** $\mathcal{E}((N,e), M)$ |
|---|---|
| $(N,e,d,p,q) \xleftarrow{\$} \mathcal{K}_{\mathrm{rsa}}$ | $Pad \xleftarrow{\$} \{0,1\}^{\ell}$ ; $x \leftarrow 0 \parallel Pad \parallel M$ |
| $\mathrm{return}\ ((N,e),(N,d))$ | $C \leftarrow x^e \bmod N$ ; $\mathrm{return}\ C$ |

1. Specify a $\mathcal{O}(k^3)$-time decryption algorithm $\mathcal{D}$ such that $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is an asymmetric encryption scheme satisfying the correct decryption property.
2. Specify an adversary $A$ making at most $2^{\ell/2}$ queries to its **LR** oracle and achieving $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{AE}}(A) \ge 1/4$. Your adversary should have $\mathcal{O}(k \cdot 2^{\ell/2})$ running time, not counting the time taken by game procedures to execute.