Public-Key Encryption

Adam O'Neill

based on http://cseweb.ucsd.edu/~mihir/cse207/

Symmetric-key Crypto

- Before Alice and Bob can communicate securely, they need to have a common secret key K_{AB}.
- If Alice wishes to also communicate with Charlie then she and Charlie must also have another common secret key K_{AC}.
- If Alice generates K_{AB}, K_{AC}, they must be communicated to her partners over private and authenticated channels.

Public-key Crypto

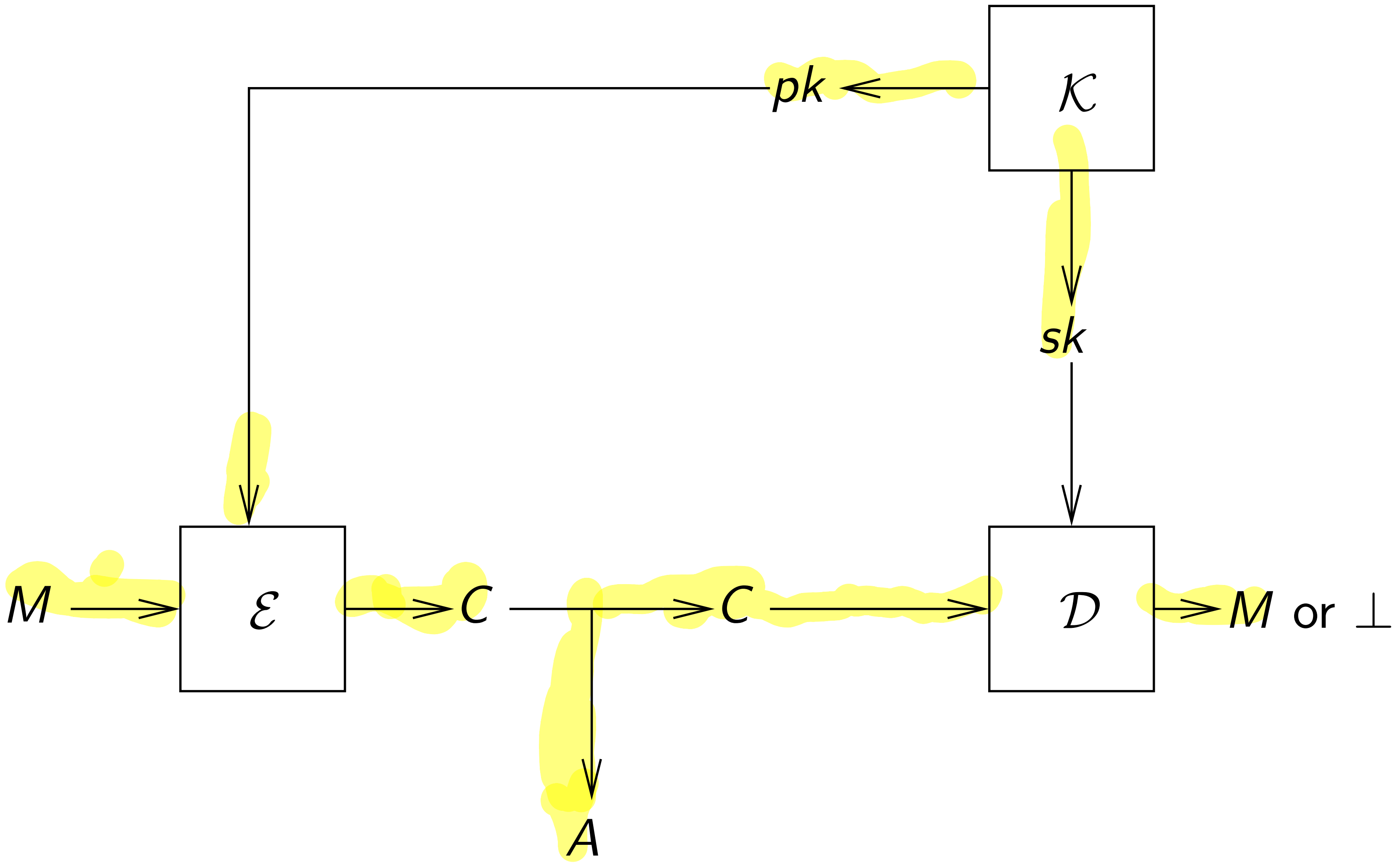
- Alice has a secret key that is shared with nobody, and an associated public key that is known to everybody.
- Anyone (Bob, Charlie, ...) can use Alice's public key to send her an encrypted message which only she can decrypt.

Think of the public key like a phone number that you can look up in a database

- Senders don't need secrets
- There are no shared secrets

Syntax

A public-key (or asymmetric) encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consists of three algorithms, where



How it Works

Step 1: Key generation
Alice locally computers $(pk, sk) \xleftarrow{\$} \mathcal{K}$ and stores sk.

Step 2: Alice enables any prospective sender to get pk.

Step 3: The sender encrypts under pk and Alice decrypts under sk.

We don't require privacy of pk but we do require authenticity: the sender should be assured pk is really Alice's key and not someone else's. One could

- Put public keys in a trusted but public "phone book", say a cryptographic DNS.
- Use certificates as we will see later.

Privacy

- The privacy notion is like IND-CPA for symmetric-key encryption, except the adversary is given the public key.

IND-CPA

Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a PKE scheme and A an adversary.

Game $\text{Left}_{\mathcal{AE}}$	Game $\text{Right}_{\mathcal{AE}}$
procedure Initialize	**procedure Initialize**
$(pk, sk) \xleftarrow{\$} \mathcal{K}$; return pk	$(pk, sk) \xleftarrow{\$} \mathcal{K}$; return pk
procedure LR(M_0, M_1)	**procedure LR**(M_0, M_1)
Return $C \xleftarrow{\$} \mathcal{E}_{pk}(M_0)$	Return $C \xleftarrow{\$} \mathcal{E}_{pk}(M_1)$

Associated to $\mathcal{AE}, A$ are the probabilities

$$\Pr\left[\text{Left}_{\mathcal{AE}}^{A} \Rightarrow 1\right] \quad \Big| \quad \Pr\left[\text{Right}_{\mathcal{AE}}^{A} \Rightarrow 1\right]$$

that A outputs 1 in each world. The ind-cpa advantage of A is

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

Finalize(b)
ret b

lossy enc.
$pk \approx pk'$
∄ dec. key

No finalize procedure means the "trivial" finalize procedure.

Explanation

The "return pk" statement in **Initialize** means the adversary A gets the public key pk as input. It does not get sk.

It can call **LR** with any equal-length messages M_0, M_1 of its choice to get back an encryption $C \xleftarrow{\$} \mathcal{E}_{pk}(M_b)$ of M_b under sk, where $b = 0$ in game $\mathrm{Left}_{\mathcal{AE}}$ and $b = 1$ in game $\mathrm{Right}_{\mathcal{AE}}$. Notation indicates encryption algorithm may be randomized.

A is not allowed to call **LR** with messages M_0, M_1 of unequal length. Any such A is considered invalid and its advantage is undefined or 0.

It outputs a bit, and wins if this bit equals b.

Building a Scheme

We would like security to result from the hardness of computing discrete logarithms.

Let the receiver's public key be g where $G = \langle g \rangle$ is a cyclic group. Let's let the encryption of x be g^x. Then

$$\underbrace{g^x}_{\mathcal{E}_g(x)} \xrightarrow{\text{hard}} x$$

so to recover x, adversary must compute discrete logarithms, and we know it can't, so are we done?

Key Encapsulation

- To build a PKE scheme it is often easier to first build what is called a key-encapsulation mechanism

Key Encapsulation

- To build a PKE scheme it is often easier to first build what is called a key-encapsulation mechanism
- A PKE scheme is then obtained by using hybrid encryption (the so-called KEM-DEM paradigm)

$c \xleftarrow{\$} \mathcal{E}(pk, m)$

Key Encapsulation

(y^e, y) $(x, x^{1/e} \bmod N)$ (X, Y) $f(x) = y$

A KEM $\mathcal{KEM} = (\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$ is a triple of algorithms

$(pk, sk) \xleftarrow{\$} \mathcal{KK}$

$(c, K) \xleftarrow{\$} \mathcal{EK}(pk)$

$K' \leftarrow \mathcal{DK}(sk, c)$

$K = K'$

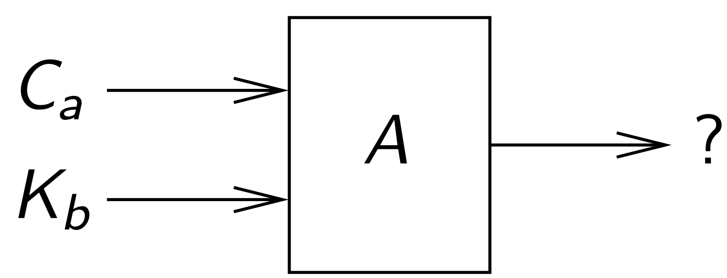
$K \in \{0,1\}^k$

KEM Security

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

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

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



A has a hard time figuring out b

KEM Security

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

Game $\text{Left}_{\mathcal{KEM}}$	Game $\text{Right}_{\mathcal{KEM}}$
procedure Initialize	**procedure Initialize**
$(pk, sk) \xleftarrow{\$} \mathcal{KK}$	$(pk, sk) \xleftarrow{\$} \mathcal{KK}$
return pk	return pk
procedure Enc	**procedure Enc**
$K_0 \xleftarrow{\$} \{0,1\}^k$; $(K_1, C_a) \xleftarrow{\$} \mathcal{EK}_{pk}$	$K_0 \xleftarrow{\$} \{0,1\}^k$; $(K_1, C_a) \xleftarrow{\$} \mathcal{EK}_{pk}$
return (K_0, C_a)	return (K_1, C_a)

We allow only one call to **Enc**. The ind-cpa advantage of A is

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

Building a KEM

$C = g^y \quad g^x = pk \quad x = sk \quad K = H(g^{xy})$

We can turn DH key exchange into a KEM via

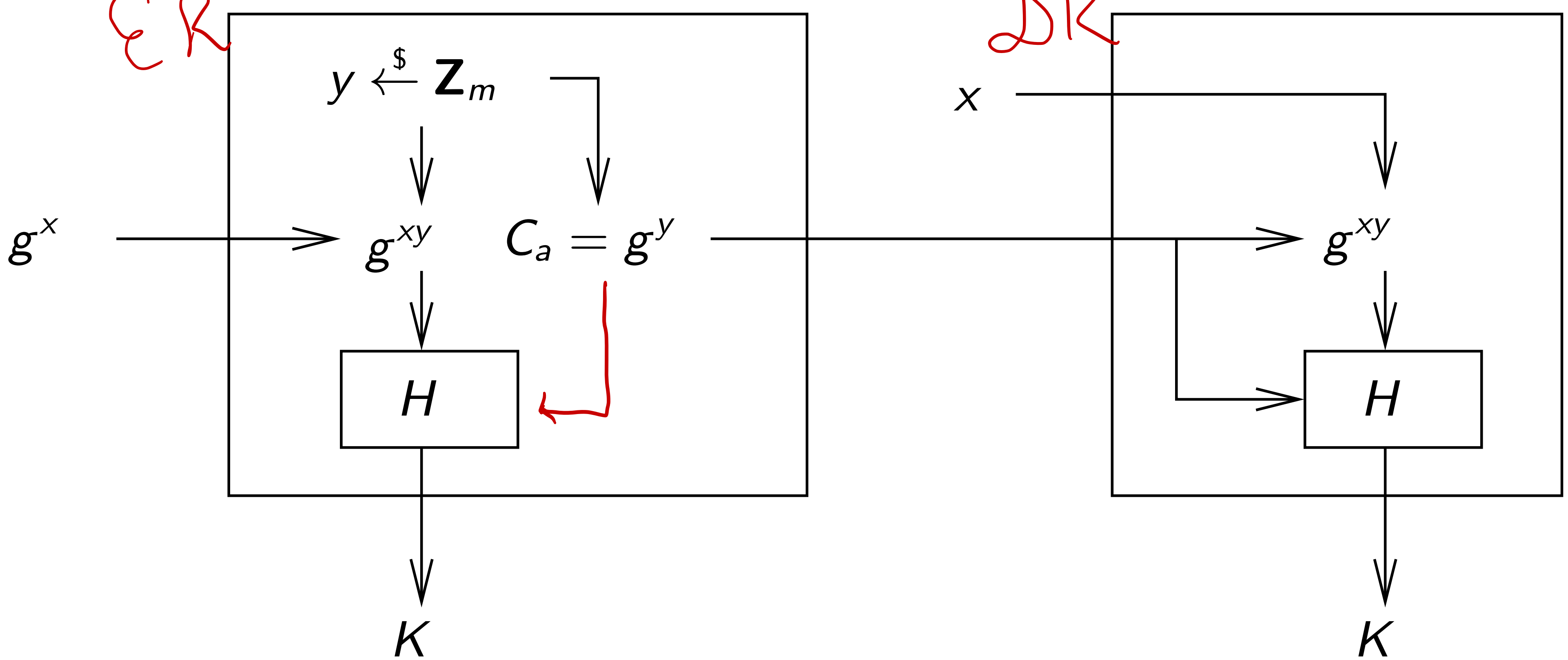
- Let Alice have public key g^x and secret key x
- Bob picks y and sends g^y to Alice as the ciphertext
- The key K is (a hash of) the shared DH key $g^{xy} = Y^x = X^y$

The DH key is a group element. Hashing results in a key that is a string of a desired length.

El Gamal KEM

Let $G = \langle g \rangle$ be a cyclic group of order m and $H : \{0,1\}^* \to \{0,1\}^k$ a (public, keyless) hash function. Define KEM $\mathcal{KEM} = (\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$ by

Alg $\mathcal{KK}$	**Alg** $\mathcal{EK}_X$	**Alg** $\mathcal{DK}_x(C_a)$
$x \xleftarrow{\$} \mathbf{Z}_m$	$y \xleftarrow{\$} \mathbf{Z}_m$; $C_a \leftarrow g^y$	$Z \leftarrow C_a^x$
$X \leftarrow g^x$	$Z \leftarrow X^y$	$K \leftarrow H(C_a \\| Z)$
return (X, x)	$K \leftarrow H(C_a \\| Z)$	return K
	return (K, C_a)	



Hybrid Encryption

Given a KEM $\mathcal{KEM} = (\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$ with key length k, we can build a PKE scheme with the aid of a symmetric encryption scheme $\mathcal{SE} = (\mathcal{KS}, \mathcal{ES}, \mathcal{DS})$ that also has key length k. Namely, define the PKE scheme $\mathcal{AE} = (\mathcal{KK}, \mathcal{E}, \mathcal{D})$ via:

Alg $\mathcal{E}_{pk}(M)$	**Alg** $\mathcal{D}_{sk}((C_a, C_s))$
$(K, C_a) \xleftarrow{\$} \mathcal{EK}_{pk}$	$K \leftarrow \mathcal{DK}_{sk}(C_a)$
$C_s \xleftarrow{\$} \mathcal{ES}_K(M)$	$M \leftarrow \mathcal{DS}_K(C_s)$
Return (C_a, C_s)	Return M

One query simplification

In assessing IND-CPA security of a PKE scheme, we may assume A makes only one **LR** query. It can be shown that this can decrease its advantage by at most the number of **LR** queries.

Theorem: Let $\mathcal{AE}$ be a PKE scheme and A an $\mathrm{ind\text{-}cpa}$ adversary making q **LR** queries. Then there is a $\mathrm{ind\text{-}cpa}$ adversary A_1 making 1 **LR** query such that

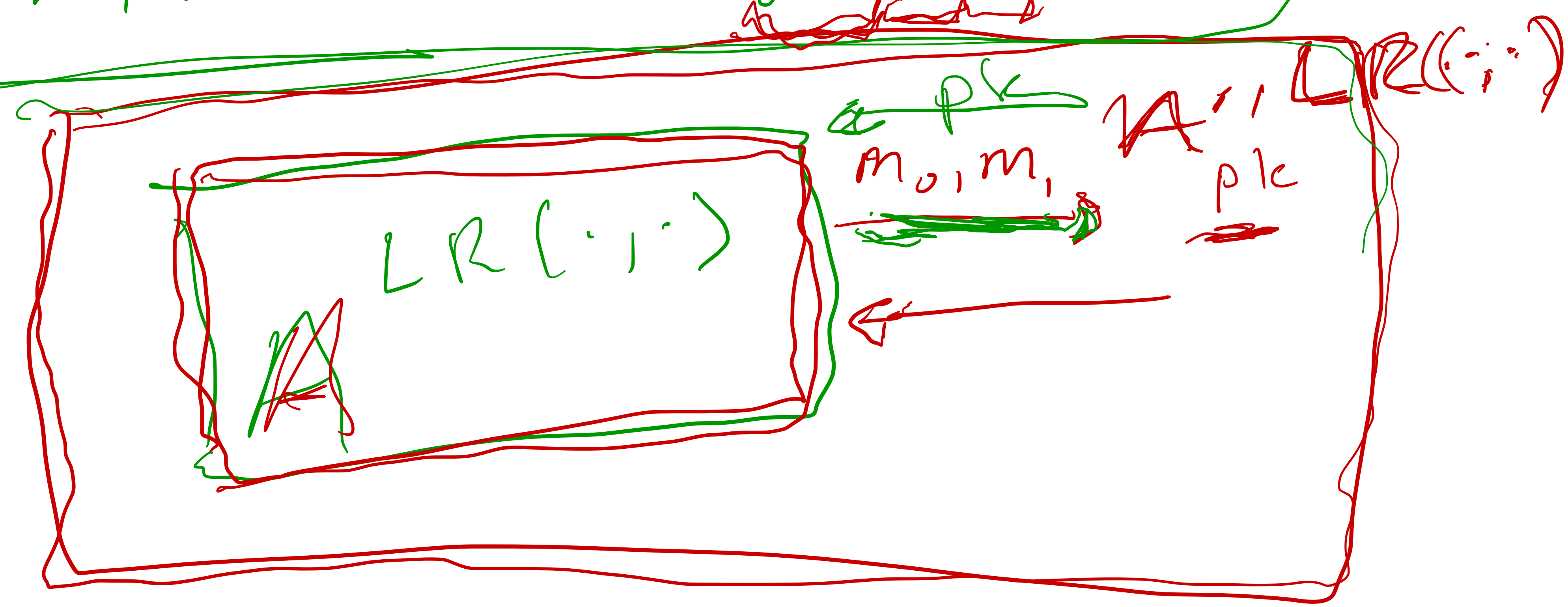
$$\mathbf{Adv}^{\mathrm{ind\text{-}cpa}}_{\mathcal{AE}}(A) \leq q \cdot \mathbf{Adv}^{\mathrm{ind\text{-}cpa}}_{\mathcal{AE}}(A_1)$$

and the running time of A_1 is about that of A.

NOT TRUE IN SYMMETRIC SETTING

Proof

q hybrids $G_0, G_1, \ldots, G_q$



i-th hybrid: queries $1, \ldots, i$ by A are answered by encryptions LEFT

$(i+1)$-st query: query own oracle

$i+2, \ldots, q$ query: encrypt RIGHT

hybrids ✓

Hybrid Encryption

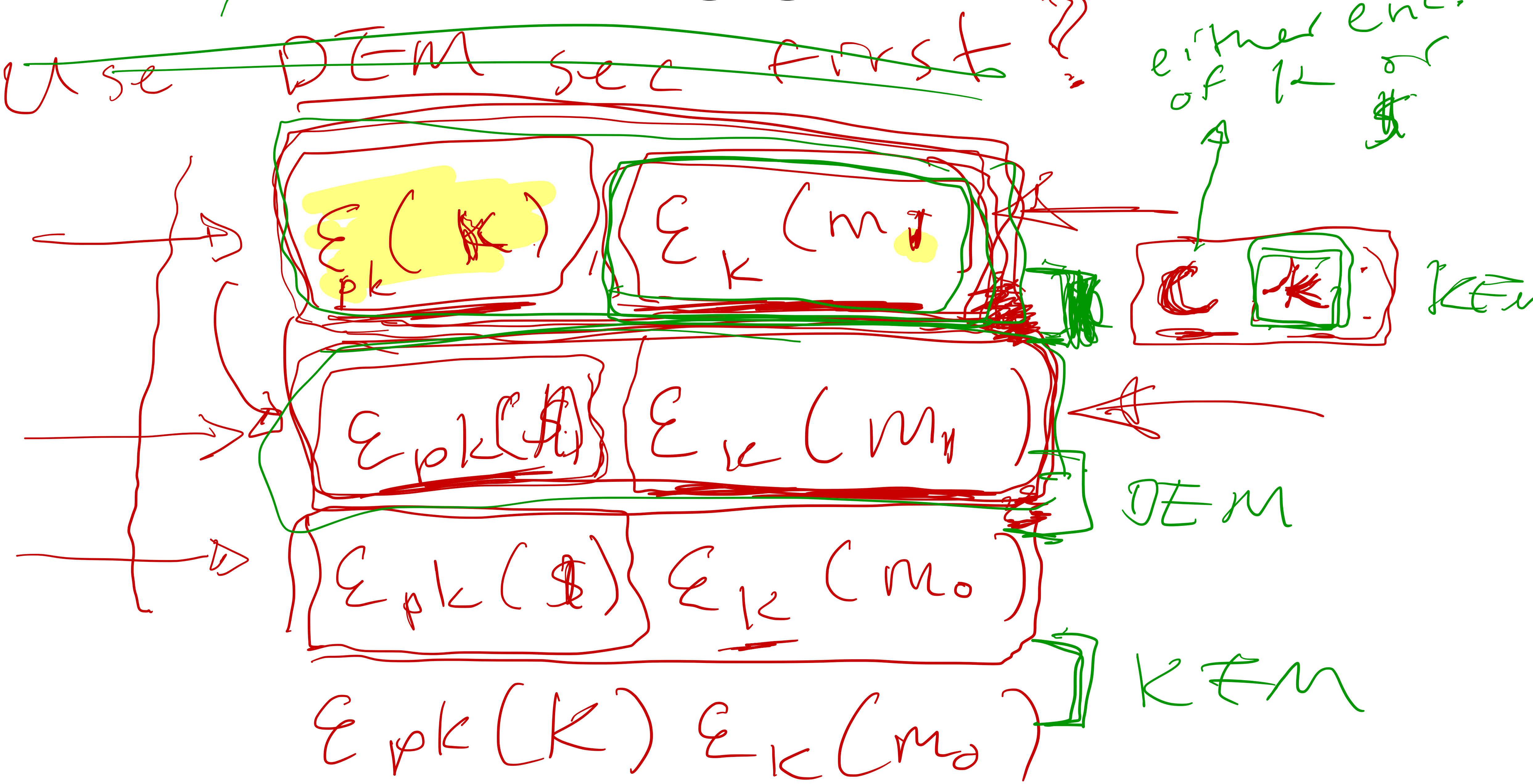
If the KEM and symmetric encryption scheme are both IND-CPA, then so is the PKE scheme constructed by hybrid encryption.

Theorem: Let KEM $\mathcal{KEM} = (\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$ and symmetric encryption scheme $\mathcal{SE} = (\mathcal{KS}, \mathcal{ES}, \mathcal{DS})$ both have key length k, and let $\mathcal{AE} = (\mathcal{KK}, \mathcal{E}, \mathcal{D})$ be the corresponding PKE scheme built via hybrid encryption. Let A be an adversary making 1 **LR** query. Then there are adversaries B_a, B_s such that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{AE}}(A) \leq 2 \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{KEM}}(B_a) + \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(B_s)\ .$$

Furthermore B_a makes one **Enc** query, B_s makes one **LR** query, and both have running time about the same as that of A.

Proof



Benefits

- Modular design, assurance via proof

Benefits

- Modular design, assurance via proof
- Speed: 160-bit elliptic curve exponentiation takes the time of about 3k-4k block cipher operations or hashes

El Gamal KEM

Let $G = \langle g \rangle$ be a cyclic group of order m and $H : \{0,1\}^* \to \{0,1\}^k$ a (public, keyless) hash function. Define KEM $\mathcal{KEM} = (\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$ by

Alg $\mathcal{KK}$	**Alg** $\mathcal{EK}_X$	**Alg** $\mathcal{DK}_x(C_a)$
$x \xleftarrow{\$} \mathbf{Z}_m$	$y \xleftarrow{\$} \mathbf{Z}_m;\ C_a \leftarrow g^y$	$Z \leftarrow C_a^x$
$X \leftarrow g^x$	$Z \leftarrow X^y$	$K \leftarrow H(C_a \| Z)$
$\mathrm{return}\ (X, x)$	$K \leftarrow H(C_a \| Z)$	$\mathrm{return}\ K$
	$\mathrm{return}\ (K, C_a)$	

How to prove this scheme is secure?

Random Oracle Model

A random oracle is a publicly-accessible random function

$\xrightarrow{W}$

$\xleftarrow{H(W)}$

If $H[W] = \perp$ then
  $H[W] \xleftarrow{\$} \{0,1\}^k$
Return $H[W]$

Oracle access to H provided to

- all scheme algorithms
- the adversary

The only access to H is oracle access.

ROM EG KEM

Let $G = \langle g \rangle$ be a cyclic group of order m and H the random oracle. Define the Random Oracle Model (ROM) KEM $\mathcal{KEM} = (\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$ by

Alg $\mathcal{KK}$	**Alg** $\mathcal{EK}_X^H$	**Alg** $\mathcal{DK}_x^H(C_a)$
$x \stackrel{\$}{\leftarrow} \mathbf{Z}_m$	$y \stackrel{\$}{\leftarrow} \mathbf{Z}_m;\ C_a \leftarrow g^y$	$Z \leftarrow C_a^x$
$X \leftarrow g^x$	$Z \leftarrow X^y$	$K \leftarrow H(C_a \| Z)$
return (X, x)	$K \leftarrow H(C_a \| Z)$	return K
	return (K, C_a)	

Algorithms $\mathcal{EK}, \mathcal{DK}$ have oracle access to the random oracle H.

ROM KEM Security

Let $\mathcal{KEM} = (\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$ be a ROM KEM with key length k, and let A be an adversary.

Game $\mathrm{INDCPA}_{\mathcal{KEM}}$	**procedure** $H(W)$
procedure Initialize	if $H[W] = \bot$ then $H[W] \xleftarrow{\$} \{0,1\}^k$
$(pk, sk) \xleftarrow{\$} \mathcal{KK}$; $b \xleftarrow{\$} \{0,1\}$	return $H[W]$
return pk	**procedure Enc**
procedure Finalize(b')	$K_0 \xleftarrow{\$} \{0,1\}^k$; $(K_1, C_a) \xleftarrow{\$} \mathcal{EK}^H_{pk}$
return $(b = b')$	return (K_b, C_a)

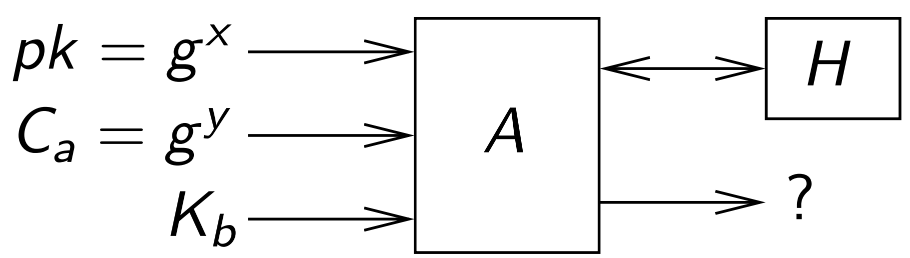
We allow only one call to **Enc**. The ind-cpa advantage of A is

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{KEM}}(A) = 2 \cdot \Pr\left[\mathrm{INDCPA}^{A}_{\mathcal{KEM}} \Rightarrow \mathsf{true}\right] - 1$$

ROM Security of EG KEM

Claim: The EG KEM is IND-CPA secure in the RO model

In the IND-CPA game



where

$$b \xleftarrow{\$} \{0,1\};\ K_0 \xleftarrow{\$} \{0,1\}^k;\ K_1 \leftarrow H(g^y \| g^{xy})$$

We are saying A has a hard time figuring out b. Why?

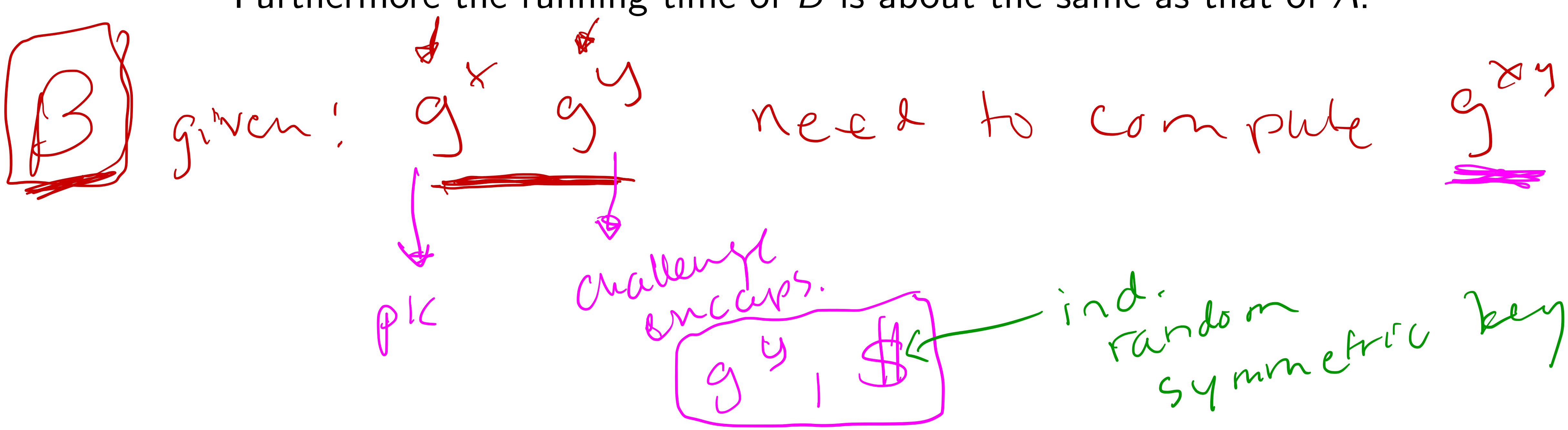
The Theorem

The following says that if the CDH problem is hard in G then the EG KEM is IND-CPA secure in the ROM.

Theorem: Let $G = \langle g \rangle$ be a cyclic group of order m and let $\mathcal{KEM} = (\mathcal{KK}, \mathcal{EK}, \mathcal{DK})$ be the ROM EG KEM over G with key length k. Let A be an ind-cpa adversary making 1 query to **Enc** and q queries to the RO H. Then there is a cdh adversary B such that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{KEM}}(A) \leq q \cdot \mathbf{Adv}^{\text{cdh}}_{G,g}(B).$$

Furthermore the running time of B is about the same as that of A.



Games for Proof

Game G_0, $\boxed{G_1}$

procedure Initialize
$x, y \xleftarrow{\$} \mathbf{Z}_m$; $K \xleftarrow{\$} \{0,1\}^k$
return g^x

procedure Enc
return (K, g^y)

procedure $H(W)$
$H[W] \xleftarrow{\$} \{0,1\}^k$; $Y||Z \leftarrow W$
if $(Z = g^{xy}$ and $Y = g^y)$ then
  bad $\leftarrow$ true; $\boxed{H[W] \leftarrow K}$
return $H[W]$

Assume (wlog) that A never repeats a H-query. Then

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{KEM}}(A) = \Pr[G_1^A \Rightarrow 1] - \Pr[G_0^A \Rightarrow 1]$$

$$\leq \Pr[G_0^A \textit{ sets } \text{bad}]$$

We would like to design B so that $\Pr[G_0^A \textit{ sets } \mathsf{bad}] \leq \mathbf{Adv}^{\mathrm{cdh}}_{G,g}(B)$

adversary $B(g^x, g^y)$

$K \stackrel{\$}{\leftarrow} \{0,1\}^k$
$b' \leftarrow A^{\mathrm{EncSim},\mathrm{HSim}}(g^x)$

subroutine EncSim
return (K, g^y)

subroutine HSim(W)
$H[W] \stackrel{\$}{\leftarrow} \{0,1\}^k$; $Y||Z \leftarrow W$
if $(Z = g^{xy}$ and $Y = g^y)$ then
  output Z and halt
return $H[W]$

Problem: B can't do the test since it does not know g^{xy}.

DHIES and ECIES

The PKE scheme derived from KEM + symmetric encryption scheme with

- The RO EG KEM
- Some suitable mode of operation symmetric encryption scheme (e.g. CBC$) is standardized as DHIES and ECIES

ECIES features:

Operation	**Cost**
encryption	2 160-bit exp
decryption	1 160-bit exp
ciphertext expansion	160-bits

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

Instantiating the RO

We have studied the EG KEM in an abstract model where H is a random function accessible only as an oracle. To get a "real" scheme we need to instantiate H with a "real" function

How do we do this securely?

Instantiating the RO

We know that PRFs approximate random functions, meaning if $F : \{0,1\}^s \times D \to \{0,1\}^k$ is a PRF then the I/O behavior of F_K is like that of a random function.

So can we instantiate H via F?

RO Paradigm

- Design and analyze schemes in RO model
- In instantiation, replace RO with a hash-function based construct.

Example: $H(W)$ = first 128 bits of SHA1(W). More generally if we need ℓ output bits:

$H(W)$ = first ℓ bits of SHA1($1||W$) || SHA1($2||W$) || $\ldots$

RO Paradigm

There is no proof that the instantiated scheme is secure based on some "standard" assumption about the hash function.

The RO paradigm is a heuristic that seems to work well in practice.

The RO model is a model, not an assumption on H. To say

"Assume SHA1 is a RO"

makes no sense: it isn't.

RO Paradigm

It yields practical, natural schemes with provable support that has held up well in practice.

Cryptanalysts will often attack schemes assuming the hash functions in them are random, and a RO proof indicates security against such attacks.

Bottom line on RO paradigm:

- Use, but use with care
- Have a balanced perspective: understand both strengths and limitations
- Research it!

Counter-Example

Let $\mathcal{AE}' = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$ be an IND-CPA PKE scheme. We modify it to a ROM PKE scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, which

- Is IND-CPA secure in the ROM, but
- Fails to be IND-CPA secure for all instantiations of the RO.

Counter-Example

Given $\mathcal{AE}' = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$ we define $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ via

Alg $\mathcal{E}^{H}_{pk}(M)$

Parse M as $\langle h \rangle$ where $h : \{0,1\}^* \to \{0,1\}^k$
$x \xleftarrow{\$} \{0,1\}^k$
$\text{if } H(x) = h(x) \text{ then return } M$
$\text{else return } \mathcal{E}'_{pk}(M)$

If H is a RO then for any $M = \langle h \rangle$

$$\Pr[H(x) = h(x)] \leq \frac{q}{2^k}$$

for an adversary making q queries to H, and hence security is hardly affected.

Counter-Example

Given $\mathcal{AE}' = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$ we define $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ via

Alg $\mathcal{E}^{H}_{pk}(M)$

Parse M as $\langle h \rangle$ where $h : \{0,1\}^* \to \{0,1\}^k$
$x \stackrel{\$}{\leftarrow} \{0,1\}^k$
$\text{if } H(x) = h(x) \text{ then return } M$
$\text{else return } \mathcal{E}'_{pk}(M)$

Now let $h : \{0,1\}^* \to \{0,1\}^k$ be any fixed function, and instantiate H with h. Then if we encrypt $M = \langle h \rangle$ we have

$$\mathcal{E}^{h}_{pk}(\langle h \rangle) = M$$

so the scheme is insecure.

Chosen Ciphertext Attack

Where we are

- We’ve seen EG KEM and extensions in the RO model

Where we are

- We've seen EG KEM and extensions in the RO model
- Besides discrete-log-based PKE schemes, the other big class of schemes is RSA-based (related to factoring)

Where we are

- We’ve seen EG KEM and extensions in the RO model
- Besides discrete-log-based PKE schemes, the other big class of schemes is RSA-based (related to factoring)
- Let’s first look at the math behind RSA

RSA Math

Recall that $\varphi(N) = |\mathbf{Z}_N^*|$.

Claim: Suppose $e, d \in \mathbf{Z}_{\varphi(N)}^*$ satisfy $ed \equiv 1 \pmod{\varphi(N)}$. Then for any $x \in \mathbf{Z}_N^*$ we have

$$(x^e)^d \equiv x \pmod{N}$$

Proof:

$$(x^e)^d \equiv x^{ed \mod \varphi(N)} \equiv x^1 \equiv x$$

modulo N

RSA Function

A modulus N and encryption exponent e define the RSA function $f : \mathbf{Z}_N^* \to \mathbf{Z}_N^*$ defined by

$$f(x) = x^e \mod N$$

for all $x \in \mathbf{Z}_N^*$.

A value $d \in Z_{\varphi(N)}^*$ satisfying $ed \equiv 1 \pmod{\varphi(N)}$ is called a decryption exponent.

Claim: The RSA function $f : \mathbf{Z}_N^* \to \mathbf{Z}_N^*$ is a permutation with inverse $f^{-1} : \mathbf{Z}_N^* \to \mathbf{Z}_N^*$ given by

$$f^{-1}(y) = y^d \mod N$$

Proof: For all $x \in \mathbf{Z}_N^*$ we have

$$f^{-1}(f(x)) \equiv (x^e)^d \equiv x \pmod{N}$$

by previous claim.

Example

Let $N = 15$. So

$$\mathbf{Z}_N^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$$
$$\varphi(N) = 8$$
$$\mathbf{Z}_{\varphi(N)}^* = \{1, 3, 5, 7\}$$

Let $e = 3$ and $d = 3$. Then

$$ed \equiv 9 \equiv 1 \pmod 8$$

Let

$$f(x) = x^3 \bmod 15$$
$$g(y) = y^3 \bmod 15$$

x	$f(x)$	$g(f(x))$
1	1	1
2	8	2
4	4	4
7	13	7
8	2	8
11	11	11
13	7	13
14	14	14

RSA Usage

- $pk = N, e;\ \ sk = N, d$
- $\mathcal{E}_{pk}(x) = x^e \mod N = f(x)$
- $\mathcal{D}_{sk}(y) = y^d \mod N = f^{-1}(y)$

Security will rely on it being hard to compute f^{-1} without knowing d.

RSA is a trapdoor, one-way permutation:

- Easy to invert given trapdoor d
- Hard to invert given only N, e

RSA Generators

An RSA generator with security parameter k is an algorithm $\mathcal{K}_{rsa}$ that returns N, p, q, e, d satisfying

- p, q are distinct odd primes
- $N = pq$ and is called the (RSA) modulus
- $|N| = k$, meaning $2^{k-1} \leq N \leq 2^k$
- $e \in \mathbf{Z}^*_{\varphi(N)}$ is called the encryption exponent
- $d \in \mathbf{Z}^*_{\varphi(N)}$ is called the decryption exponent
- $ed \equiv 1 \pmod{\varphi(N)}$

More Math

Fact: If p, q are distinct primes and $N = pq$ then $\varphi(N) = (p-1)(q-1)$.

Proof:

$$\begin{aligned}
\varphi(N) &= |\{1, \ldots, N-1\}| - |\{ip : 1 \le i \le q-1\}| - |\{iq : 1 \le i \le p-1\}| \\
&= (N-1) - (q-1) - (p-1) \\
&= N - p - q + 1 \\
&= pq - p - q + 1 \\
&= (p-1)(q-1)
\end{aligned}$$

Example:

- $15 = 3 \cdot 5$
- $\mathbf{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$
- $\varphi(15) = 8 = (3-1)(5-1)$

Building RSA Generators

Say we wish to have $e = 3$ (for efficiency). The generator $\mathcal{K}_{rsa}^{3}$ with (even) security parameter k:

repeat

$\quad p, q \xleftarrow{\$} \{2^{k/2-1}, \ldots, 2^{k/2} - 1\};\ N \leftarrow pq;\ M \leftarrow (p-1)(q-1)$

until

$\quad N \geq 2^{k-1}$ and p, q are prime and $gcd(e, M) = 1$

$d \leftarrow \text{MOD-INV}(e, M)$

return N, p, q, e, d

One-Wayness

The following should be hard:

Given: N, e, y where $y = f(x) = x^e \mod N$

Find: x

Formalism picks x at random and generates N, e via an RSA generator.

One-Wayness

Let $\mathcal{K}_{\text{rsa}}$ be a RSA generator and I an adversary.

Game $\text{OW}_{\mathcal{K}_{\text{rsa}}}$

procedure Initialize
$(N, p, q, e, d) \xleftarrow{\$} \mathcal{K}_{\text{rsa}}$
$x \xleftarrow{\$} \mathbf{Z}_N^*$; $y \leftarrow x^e \mod N$
return N, e, y

procedure Finalize(x')
return $(x = x')$

The ow-advantage of I is

$$\mathbf{Adv}^{\text{ow}}_{\mathcal{K}_{\text{rsa}}}(I) = \Pr\left[\text{OW}^{I}_{\mathcal{K}_{\text{rsa}}} \Rightarrow \text{true}\right]$$

Inverting RSA

Inverting RSA : given N, e, y find x such that $x^e \equiv y \pmod{N}$

↑ EASY — because $f^{-1}(y) = y^d \bmod N$

Know d

↑ EASY — because $d = e^{-1} \bmod \varphi(N)$

Know $\varphi(N)$

↑ EASY — because $\varphi(N) = (p-1)(q-1)$

Know p, q

↑ ?

Know N

Factoring

Given: N where $N = pq$ and p, q are prime

Find: p, q

If we can factor we can invert RSA. We do not know whether the converse is true, meaning whether or not one can invert RSA without factoring.

Factoring

Alg $\underline{\text{FACTOR}(N)}$ // $N = pq$ where p, q are primes

for $i = 2, \ldots, \lceil \sqrt{N} \rceil$ do

  if $N \bmod i = 0$ then

    $p \leftarrow i$; $q \leftarrow N/i$; return p, q

Factoring

Algorithm	**Time taken to factor N**
Naive	$O(e^{0.5 \ln N})$
Quadratic Sieve (QS)	$O(e^{c(\ln N)^{1/2}(\ln \ln N)^{1/2}})$
Number Field Sieve (NFS)	$O(e^{1.92(\ln N)^{1/3}(\ln \ln N)^{2/3}})$

Factoring

Number	bit-length	Factorization	alg
RSA-400	400	1993	QS
RSA-428	428	1994	QS
RSA-431	431	1996	NFS
RSA-465	465	1999	NFS
RSA-515	515	1999	NFS
RSA-576	576	2003	NFS
RSA-768	768	2009	NFS

Factoring

Current wisdom: For 80-bit security, use a 1024 bit RSA modulus

80-bit security: Factoring takes 2^{80} time.

Factorization of RSA-1024 seems out of reach at present.

Estimates vary, and for more security, longer moduli are recommended.

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 $x = f^{-1}(y)$

Plain RSA Encryption

The plain RSA PKE scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ associated to RSA generator $\mathcal{K}_{\mathrm{rsa}}$ is

Alg $\mathcal{K}$	**Alg** $\mathcal{E}_{pk}(M)$	**Alg** $\mathcal{D}_{sk}(C)$
$(N, p, q, e, d) \xleftarrow{\$} \mathcal{K}_{\mathrm{rsa}}$	$C \leftarrow M^e \mod N$	$M \leftarrow C^d \mod N$
$pk \leftarrow (N, e)$	return C	return M
$sk \leftarrow (N, d)$		
return (pk, sk)		

The "easy-backwards with trapdoor" property implies

$$\mathcal{D}_{sk}(\mathcal{E}_{pk}(M)) = M$$

for all $M \in \mathbf{Z}_N^*$.

RSA-KEM

The ROM SRSA (Simple RSA) KEM $\mathcal{KEM} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ associated to RSA generator $\mathcal{K}_{\rm rsa}$ is as follows, where $H : \{0,1\}^* \to \{0,1\}^k$ is the RO:

Alg $\mathcal{K}$	**Alg** $\mathcal{E}^H_{pk}$	**Alg** $\mathcal{D}^H_{sk}(C_a)$
$(N, p, q, e, d) \stackrel{\$}{\leftarrow} \mathcal{K}_{\rm rsa}$	$x \stackrel{\$}{\leftarrow} \mathbf{Z}^*_N$	$x \leftarrow C_a^d \mod N$
$pk \leftarrow (N, e)$	$K \leftarrow H(x)$	$K \leftarrow H(x)$
$sk \leftarrow (N, d)$	$C_a \leftarrow x^e \mod N$	return K
return (pk, sk)	return (K, C_a)	

RSA-KEM

Theorem: Let $\mathcal{K}_{\mathrm{rsa}}$ be a RSA generator and $\mathcal{KEM} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ the associated ROM SRSA KEM. Let A be an ind-cpa adversary that makes 1 **Enc** query and q queries to the RO H. Then there is a OW-adversary I such that

$$\mathbf{Adv}^{\mathrm{ind\text{-}cpa}}_{\mathcal{KEM}}(A) \leq \mathbf{Adv}^{\mathrm{ow}}_{\mathcal{K}_{\mathrm{rsa}}}(I)$$

Furthermore the running time of I is about that of A plus the time for q RSA encryptions.

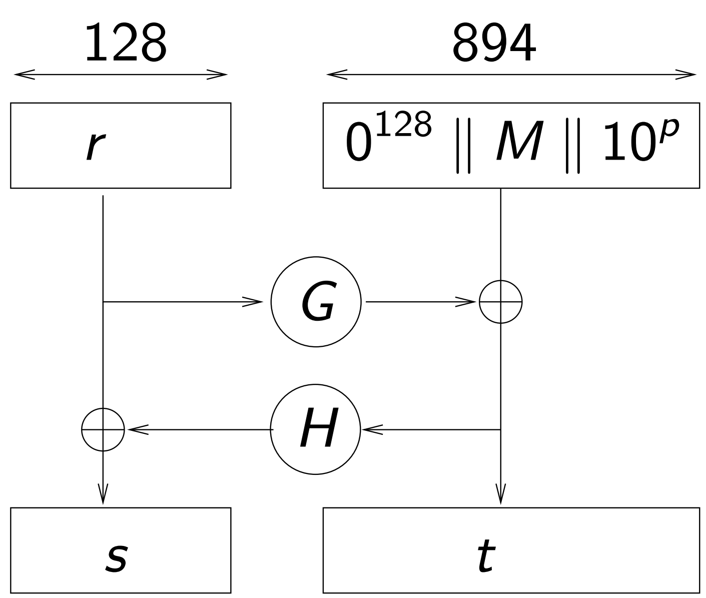
Proof

RSA-OAEP

Receiver keys: $pk = (N, e)$ and $sk = (N, d)$ where $|N| = 1024$
ROs: $G\colon \{0,1\}^{128} \to \{0,1\}^{894}$ and $H\colon \{0,1\}^{894} \to \{0,1\}^{128}$

Algorithm $\mathcal{E}_{N,e}(M)$ // $|M| \leq 765$

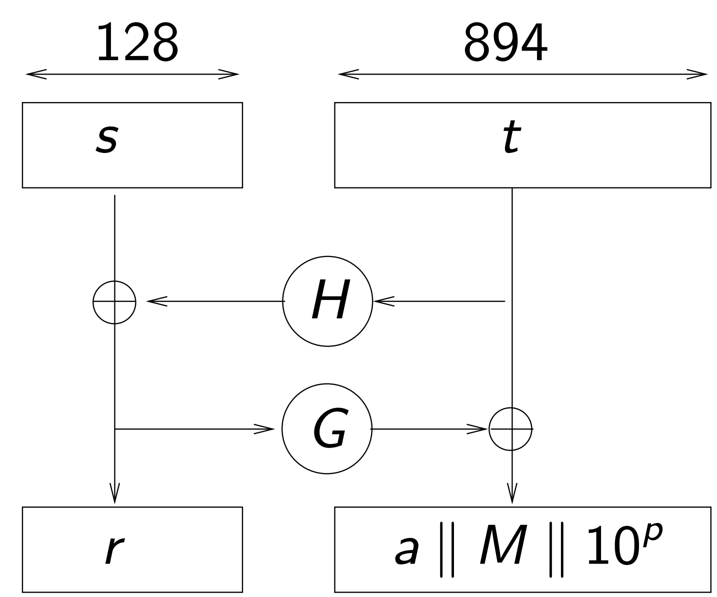
$r \stackrel{\$}{\leftarrow} \{0,1\}^{128}$; $p \leftarrow 765 - |M|$



$x \leftarrow s||t$
$C \leftarrow x^e \bmod N$
return C

Algorithm $\mathcal{D}_{N,d}(C)$ // $C \in \mathbb{Z}_N^*$

$x \leftarrow C^d \bmod N$
$s||t \leftarrow x$



if $a = 0^{128}$ **then return** M
else return $\perp$

RSA-OAEP

- IND-CPA secure in the RO model [BR'94]

RSA-OAEP

- IND-CPA secure in the RO model [BR'94]
- IND-CCA secure in the RO model [FOPS'00]

RSA-OAEP

- IND-CPA secure in the RO model [BR'94]
- IND-CCA secure in the RO model [FOPS'00]
- IND-CPA secure in the standard model assuming the phi-hiding assumption [KOS'10]

RSA-OAEP

Protocols:

- SSL ver. 2.0, 3.0 / TLS ver. 1.0, 1.1
- SSH ver 1.0, 2.0
- . . .

Standards:

- RSA PKCS #1 versions 1.5, 2.0
- IEEE P1363
- NESSIE (Europe)
- CRYPTREC (Japan)
- . . .