ASYMMETRIC (PUBLIC-KEY) ENCRYPTION

Recommended Book

Steven Levy. *Crypto*. Penguin books. 2001.

A non-technical account of the history of public-key cryptography and the colorful characters involved.

Recall Symmetric Cryptography

- 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 Encryption

- 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

Public Key Encryption

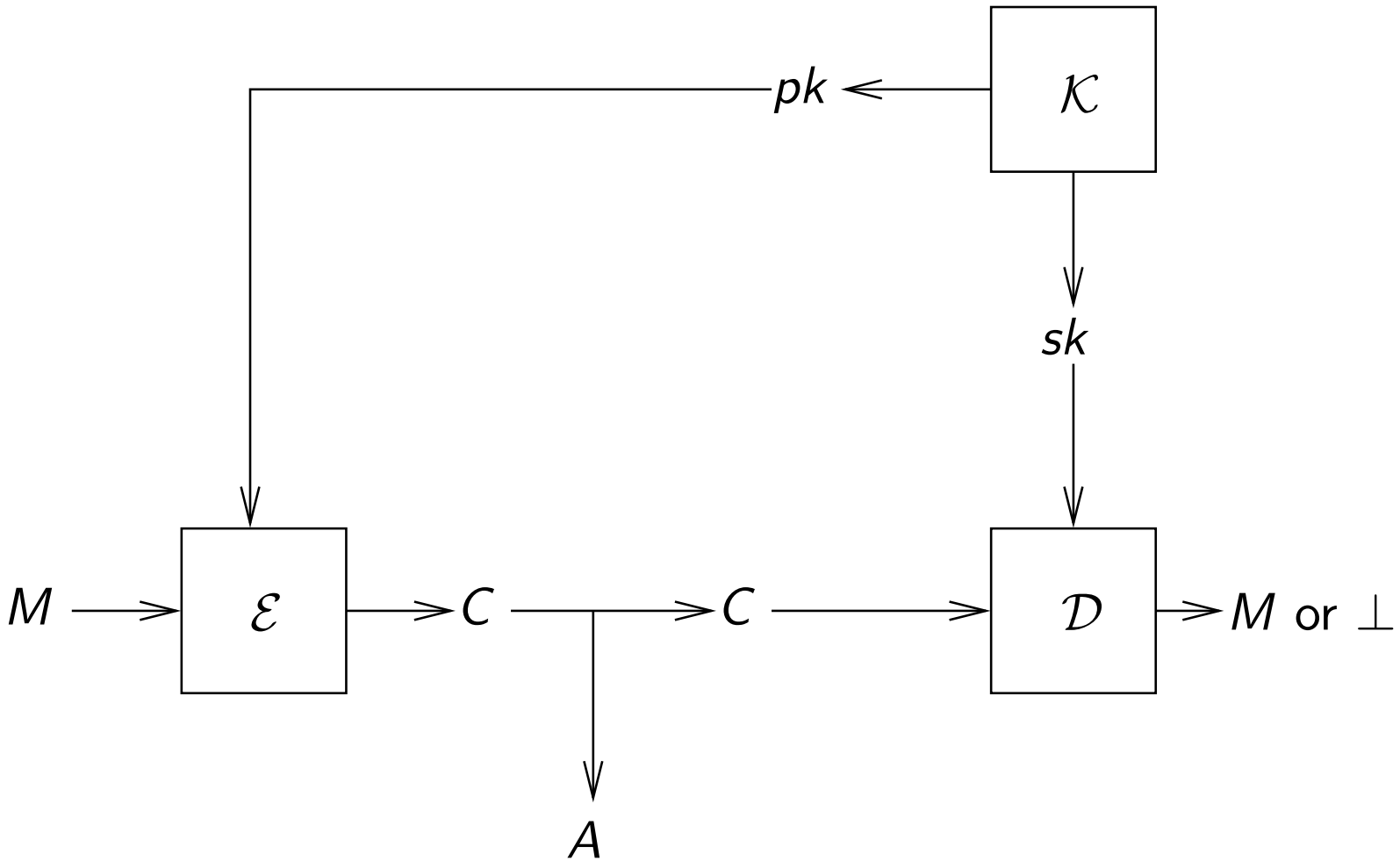
- Alice has a secret key that is shared with nobody, and an associated public key that is known to everybody.
- Anyone (Bob, Charlie, ...) can use Alice's public key to send her an encrypted message which only she can decrypt.

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 of PKE

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.

Security of PKE Schemes

Same as for symmetric encryption, except for one new element: The adversary needs to be given the public key.

We formalize IND-CPA accordingly.

The games for IND-CPA

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

Game $\mathrm{Left}_{\mathcal{AE}}$	Game $\mathrm{Right}_{\mathcal{AE}}$
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

$$\Pr\left[\mathrm{Left}^{A}_{\mathcal{AE}} \Rightarrow 1\right] \qquad \Pr\left[\mathrm{Right}^{A}_{\mathcal{AE}} \Rightarrow 1\right]$$

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

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

Building a PKE 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?

Building a PKE 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

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

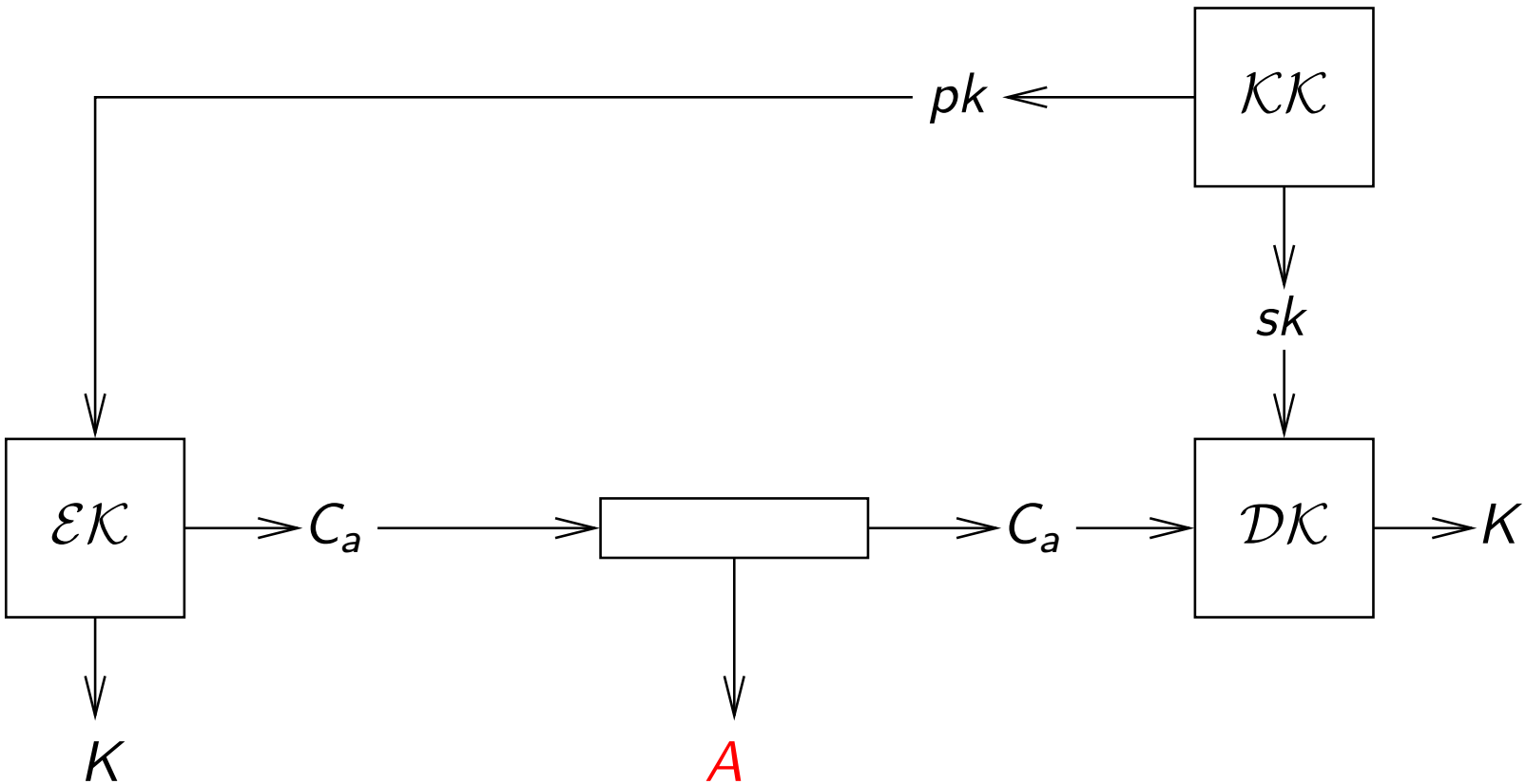
so to recover x, adversary must compute discrete logarithms, and we know it can't, so are we done?

Problem: Legitimate receiver needs to compute discrete logarithm to decrypt too! But decryption needs to be feasible.

Above, receiver has no secret key!

Key Encapsulation Mechanisms (KEMs)

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



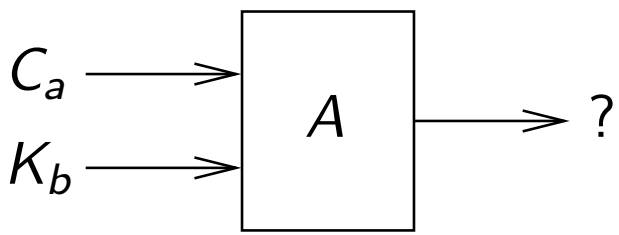
$K \in \{0,1\}^k$ is a key of some key length k associated to $\mathcal{KEM}$

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 IND-CPA security

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

Game $\text{Left}_{\mathcal{KEM}}$

procedure Initialize

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

return pk

procedure Enc

$K_0 \xleftarrow{\$} \{0,1\}^k$; $(K_1, C_a) \xleftarrow{\$} \mathcal{EK}_{pk}$

return (K_0, C_a)

Game $\text{Right}_{\mathcal{KEM}}$

procedure Initialize

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

return pk

procedure Enc

$K_0 \xleftarrow{\$} \{0,1\}^k$; $(K_1, C_a) \xleftarrow{\$} \mathcal{EK}_{pk}$

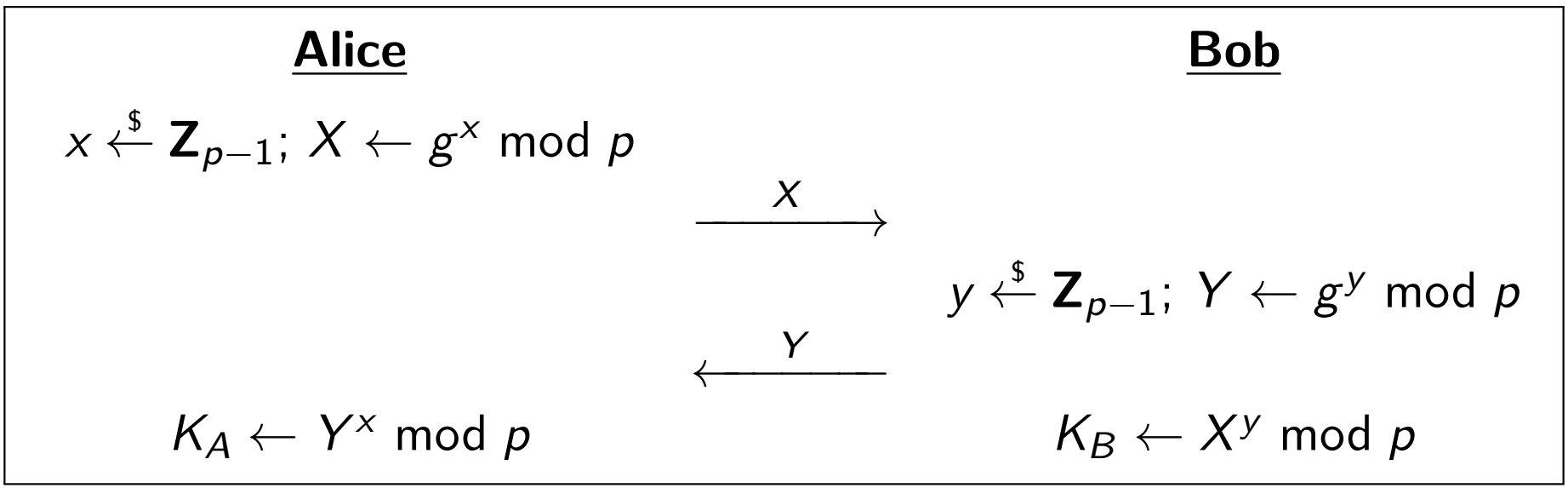
return (K_1, C_a)

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

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

Recall DH Secret Key Exchange

The following are assumed to be public: A large prime p and a generator g of $\mathbf{Z}_p^*$.



- $Y^x = (g^y)^x = g^{xy} = (g^x)^y = X^y$ modulo p, so $K_A = K_B$
- Adversary is faced with the CDH problem.

The EG KEM: Idea

We can turn DH key exchange into a KEM via

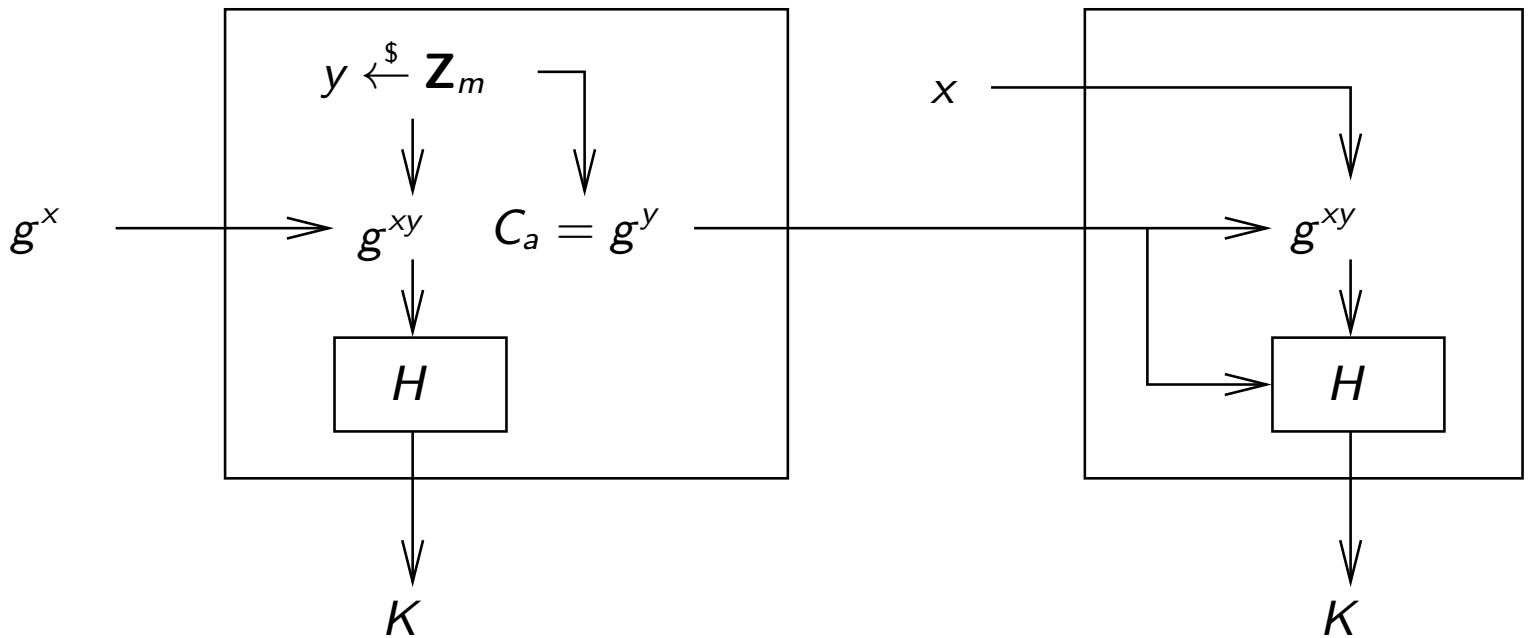
- Let Alice have public key g^x and secret key x
- Bob picks y and sends g^y to Alice as the ciphertext
- The key K is (a hash of) the shared DH key $g^{xy} = Y^x = X^y$

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

The EG KEM: Specification

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 $\mathcal{KEM} = (\mathcal{K}, \mathcal{EK}, \mathcal{DK})$ by

Alg $\mathcal{K}$	**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)	



From KEMs to PKE: 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

Simplification: For PKE we can assume just one **LR** query

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.

Hybrid encryption works

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.

Benefits of hybrid encryption

Modular design and **assurance via proof** as above.

Also **speed**.

Asymmetric cryptography is orders of magnitude slower than symmetric cryptography.

An exponentiation in a 160-bit elliptic curve group costs about the same as 3000-4000 hashes or block cipher operations.

So performance is improved by limiting the asymmetric operations as in hybrid encryption.

Proof of Theorem: Intuition

With $b \xleftarrow{\$} \{0,1\}$; $K_0 \xleftarrow{\$} \{0,1\}^k$; $(K_1, C_a) \xleftarrow{\$} \mathcal{EK}_{pk}$

Game	Challenge ciphertext	Adversary goal
G_0	$C_a, \mathcal{ES}_{K_1}(M_b)$	Compute b
G_1	$C_a, \mathcal{ES}_{K_0}(M_b)$	Compute b

- A unlikely to win in G_1 because of security of symmetric scheme
- A is about as likely to win in G_1 as in G_0 due to KEM security

Where we are

We know how to achieve PKE given

- A KEM
- A symmetric encryption scheme

We have plenty of symmetric encryption schemes, eg. AES-CTR$, AES-CBC$, ...

We need a KEM.

We have a candidate KEM, namely EG KEM.

So what remains is to evaluate the security of this KEM.

Recall: The EG 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 $\mathcal{KEM} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ by

$\underline{\textbf{Alg } \mathcal{K}}$	$\underline{\textbf{Alg } \mathcal{E}_X}$	$\underline{\textbf{Alg } \mathcal{D}_x(C_a)}$
$x \xleftarrow{\$} \mathbf{Z}_m$	$y \xleftarrow{\$} \mathbf{Z}_m;\ C_a \leftarrow g^y$	$Z \leftarrow C_a^x$
$X \leftarrow g^x$	$Z \leftarrow X^y$	$K \leftarrow H(C_a \Vert Z)$
return (X, x)	$K \leftarrow H(C_a \Vert Z)$	return K
	return (K, C_a)	

What H is suitable?

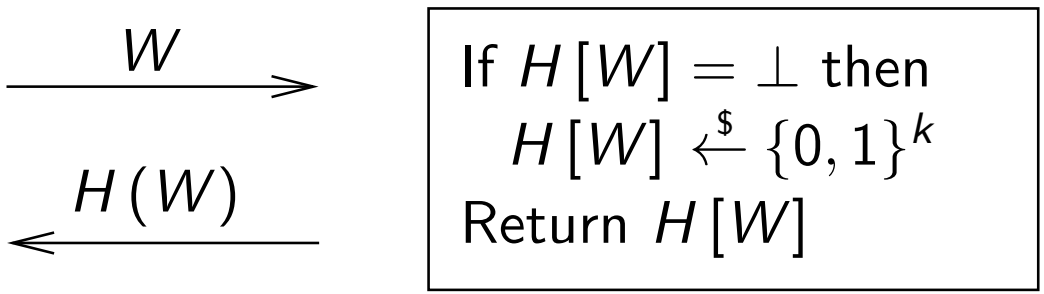
Our analysis will assume H is "perfect"

Question: What does this mean?

Answer: H will be modeled as a random oracle [BR93]

Random Oracle Model [BR93]

A random oracle is a publicly-accessible random function



Oracle access to H provided to

- all scheme algorithms
- the adversary

The only access to H is oracle access.

The 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

$$\begin{array}{l|l|l}
\underline{\textbf{Alg } \mathcal{KK}} & \underline{\textbf{Alg } \mathcal{EK}^H_X} & \underline{\textbf{Alg } \mathcal{DK}^H_x(C_a)} \\
x \xleftarrow{\$} \mathbf{Z}_m & y \xleftarrow{\$} \mathbf{Z}_m;\ C_a \leftarrow g^y & Z \leftarrow C_a^x \\
X \leftarrow g^x & Z \leftarrow X^y & K \leftarrow H(C_a \| Z) \\
\text{return } (X, x) & K \leftarrow H(C_a \| Z) & \text{return } K \\
 & \text{return } (K, C_a) &
\end{array}$$

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

ROM KEM IND-CPA 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 $(pk, sk) \xleftarrow{\$} \mathcal{KK}$; $b \xleftarrow{\$} \{0,1\}$ return pk	if $H[W] = \perp$ then $H[W] \xleftarrow{\$} \{0,1\}^k$ return $H[W]$
procedure Finalize(b') return $(b = b')$	**procedure Enc** $K_0 \xleftarrow{\$} \{0,1\}^k$; $(K_1, C_a) \xleftarrow{\$} \mathcal{EK}^H_{pk}$ return (K_b, C_a)

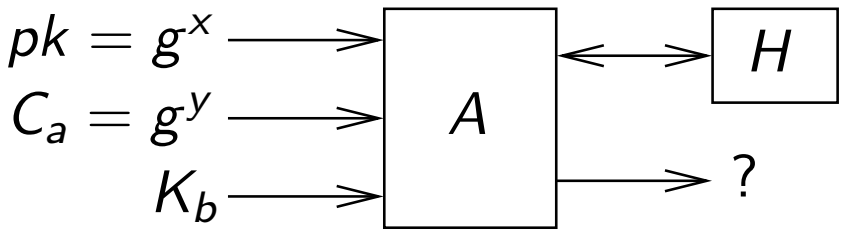
We allow only one call to **Enc**. The ind-cpa advantage of A is

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

RO model security of our EG KEM

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

In the IND-CPA game

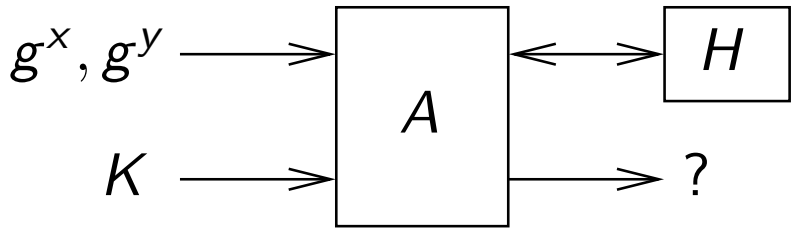


where

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

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

Intuition



where

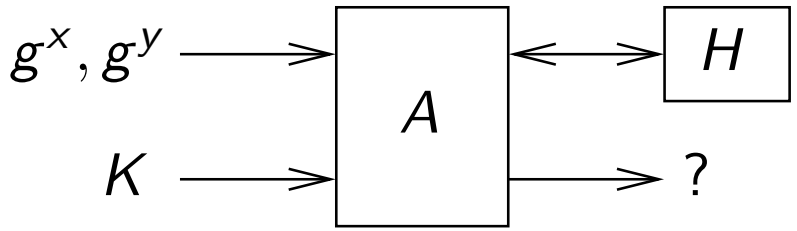
$$x, y \xleftarrow{\$} \mathbf{Z}_m;\ \ b \xleftarrow{\$} \{0,1\};\ \ K_0 \xleftarrow{\$} \{0,1\}^k;$$
$$K_1 \leftarrow H(g^y \| g^{xy});\ \ K \leftarrow K_b$$

Possible strategy for A:

- Query $g^y \| g^{xy}$ to H to get back $Z = H(g^y \| g^{xy})$
- If $Z = K$ then return 1 else return 0

This startegy works! So why do we say that A can't figure out b?

Intuition



where

$$x, y \xleftarrow{\$} \mathbf{Z}_m;\ \ b \xleftarrow{\$} \{0,1\};\ \ K_0 \xleftarrow{\$} \{0,1\}^k;$$

$$K_1 \leftarrow H(g^y \| g^{xy});\ \ K \leftarrow K_b$$

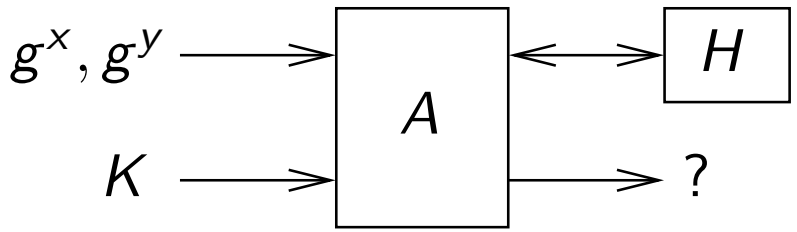
Possible strategy for A:

- Query $g^y \| g^{xy}$ to H to get back $Z = H(g^y \| g^{xy})$
- If $Z = K$ then return 1 else return 0

This startegy works! So why do we say that A can't figure out b?

Problem: A can't compute g^{xy} hence can't make the query

Intuition



where

$$x, y \xleftarrow{\$} \mathbf{Z}_m;\ \ b \xleftarrow{\$} \{0,1\};\ \ K_0 \xleftarrow{\$} \{0,1\}^k;$$

$$K_1 \leftarrow H(g^y \| g^{xy});\ \ K \leftarrow K_b$$

Observation:

- If A does not query $g^y \| g^{xy}$ to H then it cannot predict $H(g^y \| g^{xy})$ and hence has no chance at all to determine whether $K = H(g^y \| g^{xy})$ or K is random
- If A does query $g^y \| g^{xy}$ to H it has solved the CDH problem

ROM security of EG KEM

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

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

Bounding the probability of setting bad

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

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

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

subroutine EncSim
return (K, g^y)

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

Bounding the probability of setting bad

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

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

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

subroutine EncSim
return (K, g^y)

subroutine $\mathrm{HSim}(W)$
$H[W] \xleftarrow{\$} \{0,1\}^k$; $Y||Z \leftarrow W$
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}.

The generalized CDH problem

Let $G = \langle g \rangle$ be a cyclic group of order m and B' an adversary that has q outputs.

Game $\mathrm{CDH}_{G,g}$	**procedure Finalize**$(Z_1, \ldots, Z_q)$
procedure Initialize	for $i = 1, \ldots, q$ do
$x, y \xleftarrow{\$} \mathbf{Z}_m$	if $Z_i = g^{xy}$ then win $\leftarrow$ true
return g^x, g^y	return win

The cdh-advantage of B' is

$$\mathbf{Adv}^{\mathrm{cdh}}_{G,g}(B') = \Pr[\mathrm{CDH}^{B'}_{G,g} \Rightarrow \mathsf{true}]$$

Reducing generalized CDH to CDH

Lemma: Let $G = \langle g \rangle$ be a cyclic group and B' a cdh-adversary that has q outputs. Then there is a cdh-adversary B that has 1 output, about the same running time as B', and

$$\mathbf{Adv}^{\mathrm{cdh}}_{G,g}(B') \leq q \cdot \mathbf{Adv}^{\mathrm{cdh}}_{G,g}(B)$$

Proof:

$$\begin{array}{l} \underline{\textbf{adversary } B(g^x, g^y)} \\ (Z_1, \ldots, Z_q) \xleftarrow{\$} B'(g^x, g^y) \\ i \xleftarrow{\$} \{1, \ldots, q\} \\ \text{return } Z_i \end{array}$$

Bounding the probability of setting bad

We design a q-output cdh adversary B' so that

$$\Pr[G_0^A \textit{ sets } \mathsf{bad}] \leq \mathbf{Adv}^{\mathrm{cdh}}_{G,g}(B')$$

adversary $B'(g^x, g^y)$	**subroutine** EncSim
$K \xleftarrow{\$} \{0,1\}^k$	return (K, g^y)
$i \leftarrow 0$	
$b' \leftarrow A^{\mathrm{EncSim},\mathrm{HSim}}(g^x)$	**subroutine** $\mathrm{HSim}(W)$
return $Z_1, \ldots, Z_q$	$H[W] \xleftarrow{\$} \{0,1\}^k$; $Y \| Z \leftarrow W$
	$i \leftarrow i+1$; $Z_i \leftarrow Z$
	return $H[W]$

Then the cdh-adversary B of the theorem is obtained by applying the lemma to B'.

DHIES and ECIES [ABR]

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)

But what about H?

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?

PRF-based 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?

PRF-based 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?

F_K depends on a key K. Who will have K? Since the sender needs to be able to encrypt given just pk, we need to put K in pk.

Problem: The adversary has pk and PRFs don't preserve security when the key is known to the adversary.

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$(2||W)$ || ...

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.

Instantiating ROs

There are schemes which are

- Secure in the ROM
- But insecure for all instantiations of the RO by real (families of) functions.

However, these counter-example schemes are all artificial, contrived to fail.

So far it seems that the RO paradigm works (yields secure instantiated schemes) for "real and natural" schemes.

But there is no proof of this.

Why the 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!

A 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.

Programs are strings, and vice versa

Any (computable) function $H : \{0,1\}^* \to \{0,1\}^k$ has a string representation as a program $\langle H \rangle$.

Any string S can be parsed as the representation of a program P.

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 $h : \{0,1\}^* \to \{0,1\}^k$
$x \xleftarrow{\$} \{0,1\}^k$
if $H(x) = h(x)$ then return M
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 $h : \{0,1\}^* \to \{0,1\}^k$
$x \xleftarrow{\$} \{0,1\}^k$
if $H(x) = h(x)$ then return M
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.

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

The 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

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

Example

Let $N = 15$. So

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

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

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

Let

$$\begin{aligned} f(x) &= x^3 \bmod 15 \\ g(y) &= y^3 \bmod 15 \end{aligned}$$

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
- $ed \equiv 1 \pmod{\varphi(N)}$

Plan

- Building RSA generators
- Basic RSA security
- Encryption with RSA

Some more math

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

Proof:

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

Example:

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

Recall

Given $\varphi(N)$ and $e \in \mathbf{Z}^*_{\varphi(N)}$, we can compute $d \in \mathbf{Z}^*_{\varphi(N)}$ satisfying $ed \equiv 1 \pmod{\varphi(N)}$ via

$$d \leftarrow \text{MOD-INV}(e, \varphi(N)).$$

We have algorithms to efficiently test whether a number is prime, and a random number has a pretty good chance of being a prime.

Building RSA generators

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

repeat
  $p, q \xleftarrow{\$} \{2^{k/2-1}, \ldots, 2^{k/2} - 1\}$; $N \leftarrow pq$; $M \leftarrow (p-1)(q-1)$
until
  $N \geq 2^{k-1}$ 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 of RSA

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 of RSA, formally

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

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

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

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

The ow-advantage of I is

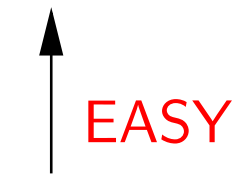
$$\mathbf{Adv}^{\mathrm{ow}}_{\mathcal{K}_{\mathrm{rsa}}}(I) = \Pr\left[\mathrm{OW}^{I}_{\mathcal{K}_{\mathrm{rsa}}} \Rightarrow \mathsf{true}\right]$$

Inverting RSA

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

Inverting RSA

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

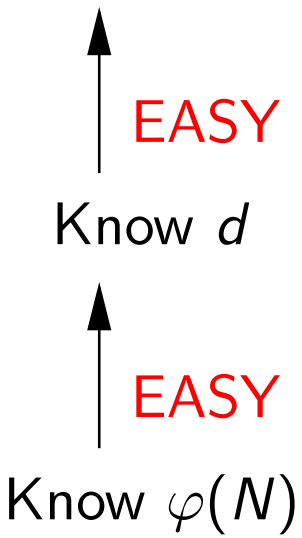


because $f^{-1}(y) = y^d \bmod N$

Know d

Inverting RSA

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



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

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

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 $d = e^{-1} \bmod \varphi(N)$

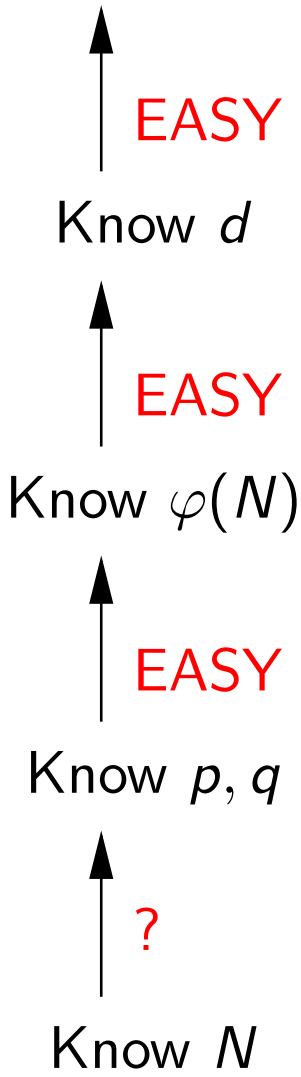
Know $\varphi(N)$

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

Know p, q

Inverting RSA

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



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

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

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

Factoring Problem

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.

A factoring algorithm

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

for $i = 2, \ldots, \left\lceil \sqrt{N} \right\rceil$ do
  if $N \bmod i = 0$ then
    $p \leftarrow i$; $q \leftarrow N/i$; return p, q

This algorithm works but takes time

$$\mathcal{O}(\sqrt{N}) = \mathcal{O}(e^{0.5 \ln N})$$

which is prohibitive.

Factoring algorithms

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 records

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

How big is big enough?

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 $\mathcal{K}_{\text{rsa}}$ is

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

The "easy-backwards with trapdoor" property implies

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

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

Plain-RSA encryption security

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

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

Getting sk from pk involves factoring N.

Plain-RSA encryption security

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

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

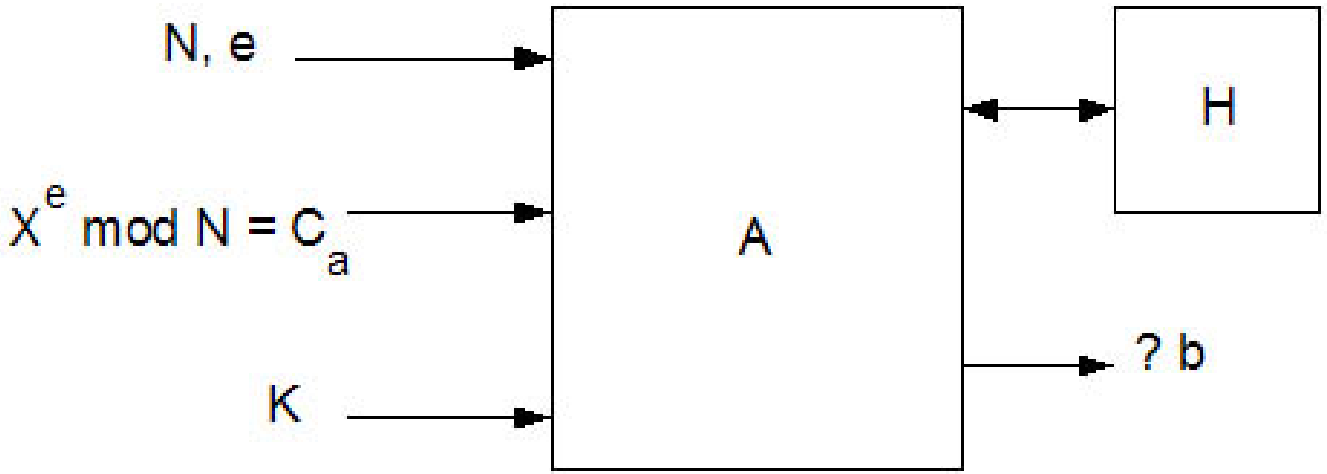
But $\mathcal{E}$ is deterministic so we can detect repeats and the scheme is not IND-CPA secure.

The ROM SRSA KEM

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

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

KEM security: Intuition



Here $x \xleftarrow{\$} \mathbf{Z}_N^*$; $b \xleftarrow{\$} \{0,1\}$; $K_0 \xleftarrow{\$} \{0,1\}^k$; $K_1 = H(x)$; $K \leftarrow K_b$;
If A queries x to H it can get $H(x)$ and test whether $K = H(x)$, but

- To find x it must invert RSA at C_a
- Without querying x it has 0 advantage in determining b
- If it queries x we can "see" this and invert RSA

SRSA KEM security: Result

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

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

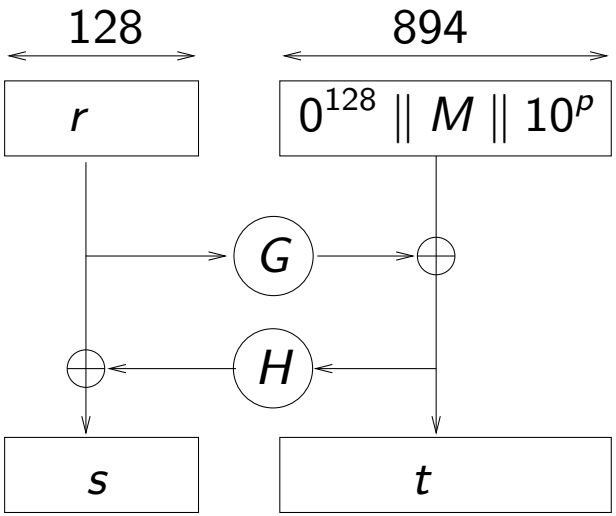
Furthermore the running time of I is about that of A plus the time for q RSA encryptions.

OAEP [BR94]

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

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

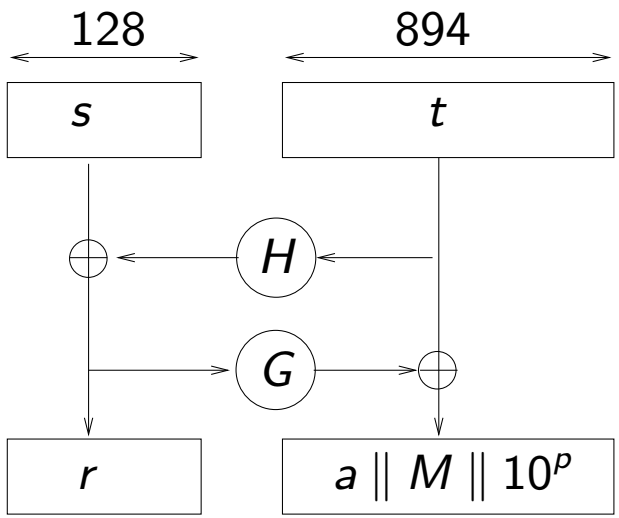
$r \xleftarrow{\$} \{0,1\}^{128}$; $p \leftarrow 765 - |M|$



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

If RSA is 1-way and H, G are random oracles then

- OAEP is IND-CPA secure [BR94]
- OAEP is IND-CCA secure [FOPS00]

RSA OAEP usage

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)
- ...