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

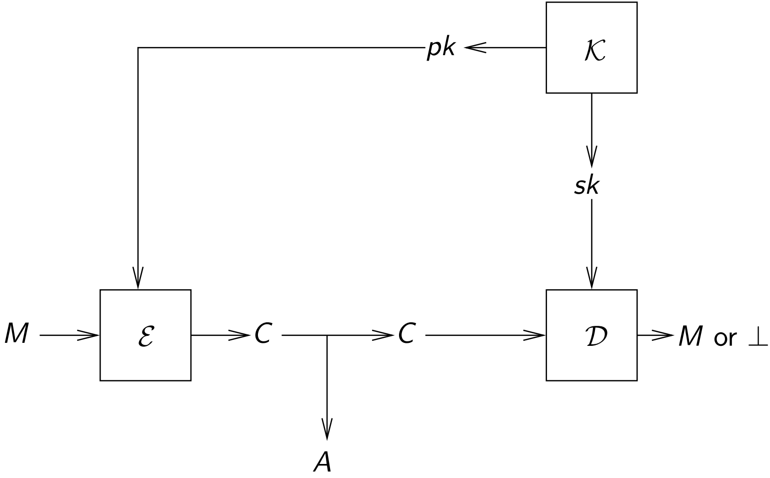
- Alice has a secret key that is shared with nobody, and an associated public key that is known to everybody.
- Anyone (Bob, Charlie, ...) can use Alice's public key to send her an encrypted message which only she can decrypt.

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



Correct decryption requirement

Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an asymmetric encryption scheme. The correct decryption requirement is that

$$\Pr[\mathcal{D}(sk, \mathcal{E}(pk, M)) = M] = 1$$

for all (pk, sk) that may be output by $\mathcal{K}$ and all messages M in the *message space* of $\mathcal{AE}$. The probability is over the random choices of $\mathcal{E}$.

This simply says that decryption correctly reverses encryption to recover the message that was encrypted. When we specify schemes, we indicate what is the message space.

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

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

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

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

IND-CPA: Explanations

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 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!

Recall DH Secret Key Exchange

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

Alice		**Bob**
$x \xleftarrow{\$} \mathbb{Z}_{p-1}$; $X \leftarrow g^x \bmod p$		
	$\xrightarrow{X}$	
		$y \xleftarrow{\$} \mathbb{Z}_{p-1}$; $Y \leftarrow g^y \bmod p$
	$\xleftarrow{Y}$	
$K_A \leftarrow Y^x \bmod p$		$K_B \leftarrow X^y \bmod p$

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

From key exchange to PKE

We can turn DH key exchange into a public key encryption scheme via

- Let Alice have public key g^x and secret key x
- If Bob wants to encrypt M for Alice, he
  - Picks y and sends g^y to Alice
  - Encrypts M under $g^{xy} = (g^x)^y$ and sends ciphertext to Alice.
- But Alice can recompute $g^{xy} = (g^y)^x$ because
  - g^y is in the received ciphertext
  - x is her secret key

Thus she can decrypt and adversary is still faced with CDH .

The DHIES scheme

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

$\underline{\textbf{Alg } \mathcal{K}}$	$\underline{\textbf{Alg } \mathcal{E}_X(M)}$	$\underline{\textbf{Alg } \mathcal{D}_x(Y, W)}$
$x \xleftarrow{\$} \mathsf{Z}_m$ $X \leftarrow g^x$ return (X, x)	$y \xleftarrow{\$} \mathsf{Z}_m$; $Y \leftarrow g^y$ $K \leftarrow X^y$ $W \leftarrow H(K) \oplus M$ return (Y, W)	$K \leftarrow Y^x$ $M \leftarrow H(K) \oplus W$ return M

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

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

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.

In practice, $H(K)$ could be the first k bits of the sequence

$$\mathsf{SHA256}(0^8 \| K) \| \mathsf{SHA256}(0^7 1 \| K) \| \cdots$$

ECIES

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

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)

Exercise

Let $p \geq 3$ be a prime, $g \in \mathrm{Z}_p^*$ a generator of Z_p^* and $H\colon\ G \to \{0,1\}^k$ a hash function. (These are all public.) Consider the key-generation and encryption algorithms below, where $M \in \{0,1\}^k$:

<u>**Alg** $\mathcal{K}$</u>	<u>**Alg** $\mathcal{E}(X, M)$</u>
$x \xleftarrow{\$} \mathrm{Z}_{p-1}^*$	$y \xleftarrow{\$} \mathrm{Z}_{p-1}$; $Y \leftarrow g^y \bmod p$
$X \leftarrow g^x \bmod p$	$Z \leftarrow X^y \bmod p$; $W \leftarrow H(Y) \oplus M$
$\mathrm{return}\ (X, x)$	Return (Z, W)

Specify a $\mathcal{O}(|p|^3 + k)$-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, and prove this is the case.

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:

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

modulo N

The RSA function

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

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

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

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

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

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

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

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

by previous claim.

Example

Let $N = 15$. So

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

Example

Let $N = 15$. So

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

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

Exercise

1. List all possible encryption exponents for RSA modulus 35:

$\boxed{}$

2. The decryption exponent corresponding to RSA modulus 187 and encryption exponent 107 is

$\boxed{}$

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 $2^{k-1} \leq N \leq 2^k$
- $e \in Z^*_{\varphi(N)}$ is called the encryption exponent
- $d \in Z^*_{\varphi(N)}$ is called the decryption exponent
- $ed \equiv 1 \pmod{\varphi(N)}$

Plan

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

A formula for Phi

Fact: Suppose $N = pq$ for distinct primes p and q. Then

$$\varphi(N) = (p-1)(q-1) \; .$$

Example: Let $N = 15 = 3 \cdot 5$. Then the Fact says that

$$\varphi(15) = (3-1)(5-1) = 8$$

. As a check, $Z_{15}^{*} = \{1, 2, 4, 7, 8, 11, 13, 14\}$ indeed has size 8.

A more general formula for Phi

Fact: Suppose $N \geq 1$ factors as

$$N = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_n^{\alpha_n}$$

where $p_1 < p_2 < \ldots < p_n$ are primes and $\alpha_1, \ldots, \alpha_n \geq 1$ are integers. Then

$$\varphi(N) = p_1^{\alpha_1 - 1}(p_1 - 1) \cdot p_2^{\alpha_2 - 1}(p_2 - 1) \cdot \ldots \cdot p_n^{\alpha_n - 1}(p_n - 1) \; .$$

Note prior Fact is a special case of the above. (Make sure you understand why!)

Example: Let $N = 45 = 3^2 \cdot 5^1$. Then the Fact says that

$$\varphi(45) = 3^1(3-1) \cdot 5^0(5-1) = 24$$

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 $\mathcal{K}_{rsa}^{3}$ with (even) security parameter k:

$$
\begin{array}{l}
\text{repeat} \\
\quad p, q \stackrel{\$}{\leftarrow} \{2^{k/2-1}, \ldots, 2^{k/2}-1\};\ N \leftarrow pq;\ M \leftarrow (p-1)(q-1) \\
\text{until} \\
\quad N \geq 2^{k-1} \text{ and } p, q \text{ are prime and } gcd(e, M) = 1 \\
d \leftarrow \text{MOD-INV}(e, M) \\
\text{return } N, p, q, e, d
\end{array}
$$

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

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

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 $x^e \equiv y \pmod{N}$

$$\begin{array}{c} \text{Inverting RSA} \\ \big\uparrow {\color{red}\text{EASY}} \\ \text{Know } d \end{array}$$

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

Inverting RSA

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

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

Know d

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

Know $\varphi(N)$

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 $x^e \equiv y \pmod N$

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

Know d

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

Know $\varphi(N)$

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

Know p, q

$\uparrow$?

Know N

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

http://www.youtube.com/watch?v=wXB-V_Keiu8

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

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

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

Plain-RSA encryption security

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

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

Plain-RSA encryption security

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

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

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

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

Getting *sk* from *pk* involves factoring N.

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

Exercise

Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the plain RSA asymmetric encryption scheme associated to RSA generator $\mathcal{K}_{\mathrm{rsa}}$. Specify in pseudocode an adversary A making one **LR** query and achieving $\mathsf{Adv}^{\mathrm{ind\text{-}cpa}}_{\mathcal{AE}}(A) = 1$. The messages in the **LR** query must both be in Z_N^* (assume they are encoded as strings of some common length), and the running time of A should be $\mathcal{O}(k)$, where k is the security parameter associated to $\mathcal{K}_{\mathrm{rsa}}$ and the time taken by game procedures to execute is not counted in the time of A.

Exercise

Let k be an integer and let $\mathcal{K}_{\mathrm{rsa}}$ be an RSA generator with associated security parameter $8k$. Assume that if (N, p, q, e, d) is an output of $\mathcal{K}_{\mathrm{rsa}}$ then $(p-1)/2$ and $(q-1)/2$ are primes larger than 2^{2k}. Let P denote the set of all odd primes smaller than 2^k. Consider the key-generation and encryption algorithms below, where the message M is in $\mathbb{Z}_N^*$:

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

1. Prove that $P \subseteq \mathbf{Z}^*_{\varphi(N)}$ for any (N, p, q, e, d) output by $\mathcal{K}_{\mathrm{rsa}}$.

2. Specify in pseudocode 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 condition, and prove that this is indeed the case. Your pseudocode should explicitly invoke algorithms from the list in the Computational Number Theory slides and you should use part **1.** above.

3. Specify in pseudocode an adversary A making one **LR** query and achieving $\mathsf{Adv}^{\mathrm{ind\text{-}cpa}}_{\mathcal{AE}}(A) = 1$. The messages in the **LR** query must both be in $\mathbf{Z}^*_N$, and the running time of A should be $\mathcal{O}(k)$, where the time taken by game procedures to execute is not counted in the time of A.

The SRSA scheme

Encrypt M unde $pk = N, e$ via:

- $x \stackrel{\$}{\leftarrow} \mathbb{Z}_N^*$; $C_a \leftarrow x^e \bmod N$;
- $K \leftarrow H(x)$
- $C_s \leftarrow K \oplus M$
- Ciphertext is (C_a, C_s)

Decrypt (C_a, C_S) under $sk = N, d$ via:

- $x \leftarrow C_a^d \bmod N$
- $K \leftarrow H(x)$
- $M \leftarrow C_s \oplus K$

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 $H \colon \{0,1\}^* \to \{0,1\}^k$ encrypts k-bit messages via:

<u>**Alg** $\mathcal{K}$</u>	<u>**Alg** $\mathcal{E}_{N,e}(M)$</u>	<u>**Alg** $\mathcal{D}_{N,d}(C_a, C_s)$</u>
$(N,p,q,e,d) \xleftarrow{\$} \mathcal{K}_{\mathrm{rsa}}$	$x \xleftarrow{\$} \mathbb{Z}_N^*$	$x \leftarrow C_a^d \bmod N$
$pk \leftarrow (N,e)$	$K \leftarrow H(x)$	$K \leftarrow H(x)$
$sk \leftarrow (N,d)$	$C_a \leftarrow x^e \bmod N$	$M \leftarrow C_s \oplus K$
return (pk, sk)	$C_s \leftarrow K \oplus M$	return M
	return (C_a, C_s)	

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 H: $\{0,1\}^* \to \{0,1\}^k$ can be proven IND-CPA assuming

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

In practice, $H(K)$ could be the first k bits of the sequence

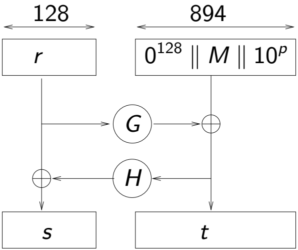
$$\mathsf{SHA256}(0^8 \| K) \| \mathsf{SHA256}(0^7 1 \| K) \| \cdots$$

OAEP [BR94]

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

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

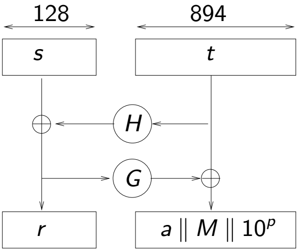
$r \xleftarrow{\$} \{0,1\}^{128}$; $p \leftarrow 765 - |M|$



$x \leftarrow s||t$
$C \leftarrow x^e \bmod N$
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 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)
- ...

Exercise

Let m, k, ℓ be integers such that $2 \le m < k$ and $k \ge 2048$ and $\ell = k - m - 1$ and ℓ 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** $\mathcal{E}((N,e),M)$
$(N,e,d,p,q) \xleftarrow{\$} \mathcal{K}_{\mathrm{rsa}}$	$Pad \xleftarrow{\$} \{0,1\}^{\ell}$; $x \leftarrow 0 \parallel Pad \parallel M$
return $((N,e),(N,d))$	$C \leftarrow x^e \bmod N$; 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 $\mathrm{Adv}^{\mathrm{ind\text{-}cpa}}_{\mathcal{AE}}(A) \ge 1/4$. Your adversary should have $\mathcal{O}(\ell \cdot 2^{\ell/2})$ running time, not counting the time taken by game procedures to execute.

PKE summary

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