CSA E0 235: Cryptography April 9 ,2015

Plain RSA and CPA-Secure RSA

Instructor: Arpita Patra *Submitted by: Abdullah.S & Pankaj K*

1 Introduction

The RSA cryptosystem, invented by Ron Rivest, Adi Shamir, and Len Adleman, is one of the practical public-key cryptosystems. It was first publicized in the August 1977 at MIT. The cryptosystem is most commonly used for providing privacy and ensuring authenticity of digital data. These days RSA is deployed in many commercial systems.

2 RSA Problem

2.1 RSA Assumption

RSA problem is based on the factoring assumption [Given N, product of two large *n-bit* primes , it is *hard* to factor N into p and q]. We define *GenRSA* as the following:

- *GenRSA* is given *N,p,q.*
- Find $\varphi(n)$ = (p-1) * (q-1).
- Choose $e > 1$, such that $gcd(\varphi, e) = 1$.
- Calculate $d := e^{-1} \bmod \varphi(n)$ i.e. $ed = 1 \bmod \varphi(n)$.

RSA experiment calculates $y = (x^e \bmod N)$. The RSA problem, informally, is to compute $[y^{1/e} \bmod N]$ for a modulus N of unknown factorization. We define the RSA experiment formally as follows:

- Run GenRSA(1^n to obtain *N,e,d*).
- Choose a uniform $y \in \mathbb{Z}^*_{\mathbb{N}}$.
- Adversary A is given *N,e,y* and it outputs $x \in \mathbb{Z}^*_{\mathbb{N}}$.
- A is successful if $x^e = y \bmod N$.

The RSA assumption is that there exists a GenRSA algorithm relative to which the RSA problem is hard. RSA problem is hard if for all PPT adversaries A, probability of success of the above experiment $\leq$ negl (n) for some negligible function n.

3 Plain RSA Encryption

Plain RSA Encryption is built on the RSA problem without any changes. We use $\langle e,N\rangle$ as the public key and $\langle d,N\rangle$ as the private key.Sender computes cipher text $c{:=}m^e \bmod N$. Receiver gets back m as $c^d \bmod N = m \bmod N$ as $ed = 1 \bmod \varphi(n)$. Encryption relies on the fact that only by knowing d, one can decrypt c. Formally we describe Plain RSA as follows :
Plain RSA:

- **Gen:** Run GenRSA(1^n to obtain N,e,d).$\langle e,N\rangle$ is the public key, pk and $\langle d,N\rangle$ is the private key, sk.
- **Enc:** For a message m and public key pk $\in \mathbb{Z}^*_{\mathbb{N}}$,Calculate $c := [m^e \bmod N]$
- **Dec:** For given ciphertext c and private key sk, compute $m := [c^d \bmod N]$.

3.1 Security of Plain RSA

Plain RSA is built on the factoring assumption. So it is computationally infeasible for any poly-time algorithms to find d and break the plain RSA. But it does not guarantee security against any other attacks by the adversary for recovering the message . Infact plain RSA is a deterministic algorithm and public key algorithms are insecure even against *CPA attack* if they are deterministic . Thus we have the following attacks that can be mounted on the plain RSA without finding d that show the weakness in the plain RSA scheme.

4 Attacks on plain RSA

4.1 Attack on small messages

If messages are small such that that the message lengths are less than $N^{1/e}$, then the ciphertext $c := [m^e \bmod N]$ becomes equal m^e. Thus the attacker finds the message by finding the e^{th} root of c. Thus for small e , it is a serious weakness . Say e=3, N is 1024 bits , then the attack works for all uniform message of around 300 bits long.

4.2 Low Public exponent Attack

It happens in practice to keep the encryption exponent e *small* so as to have faster encryptions. The most powerful attacks on low public exponent RSA are due to *Coppersmith* theorem which states

Theorem 1 *Let p(x) be a polynomial of degree e. Then in time poly(N ,e) one can find all m such that p(m) = 0 mod N and* $|m| \leq N^{1/e}$.

The theorem provides an algorithm for efficiently finding all roots of f modulo N that are less than X=$N^{1/e}$. As X gets smaller, the algorithm's running time decreases. The theorem's strength is its ability to find small roots of polynomials modulo a composite N for small e. In what follows we assume e = 3 for concreteness. Suppose the sender wants to send $m_1||m_2$ and that m_1 is known. So c = $(m_1||m_2)^e \bmod N$. Say m_2 is k bits long , attacker

defines p(x)=$(2^k.m_1$+x$)^e) - c$. This gives a e^{th} degree polynomial which has m_2 mod N as a root.

4.3 Common modulus attack

Consider a scenario where the same message is sent with the same modulus N and with different *pk,sk*. This might happen in an organisation when some supremo wants to send the same message to all his subordinates. Say we have $c_1 = m^e$ mod N_1 and $c_2 = m^e$ mod N_2 . An eavesdropper has access to the public keys N,e_1, e_2 and the encrypted messages c_1, c_2 . Since gcd$(e_1, e_2) = 1$, the eavesdropper applies the extended Euclidean algorithm to compute integers x and y such that $x * e_1 + y * e_2 = 1$. Then he computes $c_1^x * c_2^y$ mod N and this gives the message m as the above expression evaluates to $m^{e_1 * x + e_2 * y}$.

4.4 Attack on homomorphic property

Plain RSA exhibits the homomorphic property i.e. the ciphertext corresponding to the plaintext m $= m_1||m_2$ mod N is c $= c_1||c_2$ mod N.This leads to the following adaptive chosen-ciphertext attack on RSA encryption. Suppose that an active adversary wants to decrypt a particular ciphertext $c = m_e$ mod N. The adversary conceals c by selecting a random integer x $\in \mathbb{Z}_{\mathbb{N}}^*$ and computes $\bar{c} =$ cx^e mod N. Upon presentation of $\bar{c}$, challenger sends $\bar{m} = \bar{c}^d$. Since $\bar{m} = c^d(x^e)^d$, adversary gets the message m.

4.5 Iterated encryption attack

This attack is based on the Euler-Fermat theorem. On getting the ciphertext c, the adversary does the following iteratively. Keep finding $c_1 = c^e$ mod N, $c_2 = c_1^e$ mod N ... $c_k + 1 = c_k^e$ mod N. At some stage of the iteration ,if we get $c_k + 1 =$ c whick implies that $c_k = m$. This means that merely through repeating the encryption process, we eventually decrypt the originally encrypted message.

4.6 Common exponent attack

It is similar to the common modulus attack in the sense that the same message is sent to different receivers.But all receivers have the same public keys and we use a different modulus for each receiver. So the adversary sees the two different ciphertexts $c_1 = m^e$ mod n^1 and $c_2 = m^e$ mod n^2. Then an extended version of the chinese remainder theorem says that there exists a unique x $< n_1 * n_2$ (= n) such that

- x $= c_1$ mod n_1
- x $= c_2$ mod n_2

Thus the adversary is able to get m^e and he can find the e^{th} root to recover the message.

5 Final note on the attack:

There are also other clever attacks that exploit the weakness in the plain RSA. But these attacks only illustrate the pitfalls to be avoided when implementing RSA and illustrate the dangers of improper use of RSA. These can all be thwarted by a padded RSA scheme .

6 CPA-Secure Encryption

We start with describe a hard-core predicate for RSA problem and then will see how to use that hard-core predicate to encrypt the single bit.

This scheme we presenting here is less efficient to other alternative ones but for theoretical interest we are presenting the scheme.

Hard-core Predicate for the RSA Problem. Informally the RSA assumption says that given N,e, and $[x^e mod N]$ it is very hard to get back x, but it does not says anything specifically that whether it is difficult to compute any specific information about x. Now the least significant bit of x denoted by $lsb(x)$ is the hard-core predicate for the RSA problem.

The RSA hard-core predicate experiment $RSA - lsb_{A,GenRSA}(1^n)$:

1. Run $GenRSA(1^n) to obtain(N, e, d)$.
2. Choose a uniform $x \in Z_n^*$ and compute $y := [x^e mod N]$.
3. A is given N, e, y and outputs a bit b.
4. The output of the experiment is 1 iff lsb(x)=b.

Here lsb(x) is a uniform bit when $x \in Z_n^*$ is uniform.Observe that A can guess lsb(x) with probability 1/2 by simply outputting a uniform bit b.

THEOREM 1 : If the RSA problem is hard relative to GenRSA then for all probabilistic polynomial-time algorithms A there is a negligible function $negl$ such that $Pr[RSA - lsb_{A,GenRSA}(n) = 1] \leq 1/2 + negl(n)$.

We are not providing the full proof, however we provide some intuition for the theorem . We will show that to recover x from N, e, and $[x^e mod N]$ we can use some other effective algorithm A, that can calculate $lsb(r)$ while given N, e and $r^e mod N$ such that, $A([r^e mod N]) = lsb(r)$. Now we can use $A(y)$ to calculate the $lsb(x)$ while given $N, e, y = [x^e mod N]$, we can do this to obtain the bits of x. Now we need to consider two cases : **Case 1 :** $lsb(x) = 0$ This implies x is even. So we can use the naive division technique to learn about $lsb(x)$. See $y/2^e = (x/2)^e mod N$, and since x is even, so it is just the right-shift operation by one bit on x, so now $lsb(x/2)$ would be the 2nd-least significant bit of x. So by setting $y' = [y/2^e mod N]$ and then by $A(y')$ we can get second least significant bit of x.

Case 2 :

Encrypting one bit : Hardcore predicate can be used to encrypt the message $m \in 0, 1$ of single-bit length with the same spirit we explained above. We can select $r \in Z_n^*$ uniformly such that, $lsb(r) = m$, and then the ciphertext c = $[r^e mod N]$.

Construction 2 : We use GenRSA described earlier to generate (N, e, d) and then we define the public key encryption scheme as follows :

- Gen : On input 1^n run $GenRSA(1^n)$ to obtain (N, e, d). Output the public key $pk =< N, e >$, and private key $sk =< N, d >$ in tuples.
- Enc : With having $pk =< N, e >$, and single bit message m, choose $r \in Z_N^*$ uniformly such that $lsb(r) = m$ and output the cipher text $c = [r^e mod N]$.
- Dec : With having private key $sk =< N, d >$ and ciphertext c, we can compute $r = [c^d mod N]$ and output $lsb(r)$.

.

THEOREM If the RSA problem is hard relative to GenRSA then construction 2 is CPA-secure.

PROOF: Let Π be the construction 2 and A be the PPT adversary in the experiment $PubK^{eav}_{A,\Pi}(n)$. If we are able to prove that in Π eavesdropper is not able to distinguish the message encryptions then Π is CPA-Secure .

So $Pr[PubK^{eav}_{A,\Pi}(n) = 1] = 1/2.Pr[A(N, e, c) = 0|cisanencryptionofm_0] + = 1/2.Pr[A(N, e, c) = 1|cisanencryptionofm_1]$.

Now in the experiment RSA-lsb by definition, we have :

$$Pr[RSA - lsb_{A,GenRSA}(n) = 1] = Pr[A(N, e, [r^e mod N]) = lsb(r)],$$
$$= 1/2.Pr[A(N, e, [r^e mod N]) = 0|lsb(r) = 0] + 1/2.Pr[A(N, e, [r^e mod N]) = 1|lsb(r) = 1].$$

We see here that it is exactly same that when we encrypting $m \in 0, 1$ withthe case while we choosing uniform r such that lsb(r)=m, so we have now:

$$Pr[PubK - A, \Pi^{eav}] \leq 1/2 + negl(n).$$