RSA Review

Amir Hossein Jabbari

November 13, 2002

1 Public-Key Cryptography

To avoid assigning a key to each pair of individuals in private-key cryptography, while maintaining network confidentiality, public-key cryptography was introduced. The idea behind a *public-key* system is that it might be possible to find a cryptosystem where it is computationally infeasible to decypher, yet not impossible.

Since encrypting and decrypting in public-key cryptosystems require excessive time and memory on regular computers, they are not extensively used for general-purpose encryption. Therefore they are used to encrypt keys for symmetric cryptosystems such as DES[1], to transmit these keys securely through the network.

2 The RSA Cryptosystem

The *RSA Cryptosystem*[3], invented by *R. Rivest, A. Shamir, and L. Adleman* in the 1970's is a public key cryptosystem based on modular exponentiation, where the public keys are pairs (e, n), consisting of an exponent e and a modulus n that is the product of two large primes; that is, $n = pq$, where p and q are large primes, so that $(e, \phi(n)) = 1$. To form a cypher text C from a given message block M:

$$C = M^e \pmod{n}$$

where $0 \leq C \leq n$.

To decrypt the cyphertext C, knowledge of the inverse of e modulo $\phi(n)$ is required. This inverse, which is usually known as d, exists because $(e, \phi(n)) = 1$. Therefore:

$$ed \equiv 1 \pmod{\phi(n)}$$
$$ed = k\phi(n) + 1$$

Now by using p we can form the plaintext block from it's cypher.

$$D(C) \equiv C^d \pmod{n}$$
$$D(C) \equiv (M^e)^d \pmod{n}$$
$$D(C) \equiv M^{ed} \pmod{n}$$
$$D(C) \equiv M^{k\phi(n)+1} \equiv (M^{\phi(n)})^k M \pmod{n}$$

[1] Data Encryption Standard

Theorem 2.1. *Fermat's Little Theorem[4].*

If p is prime and a is a positive integer with $p \nmid a$, then $a^{p-1} \equiv 1 \pmod{p}$.

Theorem 2.2. *The Chinese Remainder Theorem[4].*

Suppose $m_1, \ldots, m_r$ are pairwise relatively prime positive integers, and suppose $a_1, \ldots, a_r$ are integers. Then, the system of r congruences $x \equiv a_i \pmod{m_i} (1 \leq i \leq r)$ has a unique solution modulo $M = m_1 \times \cdots \times m_r$, which is given by

$$x = \sum_{i=1}^{r} a_i M_i y_i \bmod M,$$

where $M_i = M/m_i$ and $y_i = M_i^{-1} \bmod m_i$, for $1 \leq i \leq r$.

Now by taking *Fermat's Little Theorem* into consideration, it can be concluded:

$$\left.\begin{array}{l} M^{p-1} \equiv 1 \pmod{p} \\ (p-1) | \phi(n) \end{array}\right\} \Rightarrow M^{k\phi(n)+1} \equiv M \pmod{p}$$

Likewise:

$$\left.\begin{array}{l} M^{q-1} \equiv 1 \pmod{q} \\ (q-1) | \phi(n) \end{array}\right\} \Rightarrow M^{k\phi(n)+1} \equiv M \pmod{q}$$

Together these last two equations and *Chinese Remainder Theorem* imply:

$$D(C) \equiv (M^{\phi(n)})^k M \equiv M \pmod{n}$$

2.1 How to choose a private key and compute a public key

Private key d should be first, a large number that a cryptanalyst cannot find by direct search and second, relatively prime to $\phi(n)$. One way of finding a private key, relatively prime to $\phi(n)$ is to choose any prime number greater than $max(p, q)$.

To find public key e, we use Euclid's algorithm, since

$$ed \equiv 1 \pmod{\phi(n)}$$

Theorem 2.3. *The Euclidean Algorithm[4].*

Let $r_0 = a$ and $r_1 = b$ be integers such that $a \geq b > 0$. If the division algorithm is successively applied to obtain $r_j = r_{j+1} q_{j+1} + r_{j+2}$, with $0 < r_{j+2} < r_{j+1}$ for $j = 0, 1, 2, \ldots, n-2$ and $r_{n+1} = 0$, then $(a, b) = r_n$, the last nonzero remainder.

Using Euclid's algorithm efficiently finds e.

2.2 How to find large prime numbers

p and q are both large prime numbers with approximately one hundred digits in practice. Finding a prime number with one hundred digits with mathematical methods takes time and effort, therefore statistical methods are used to find large prime numbers.

Theorem 2.4. *The Prime Number Theorem[4].*

The ratio of $\pi(x)$ to $x/\log x$ approaches 1 as x grows without bound. (Here, $\log x$ denotes the natural logarithm of x. In the language of limits, we have $\lim_{x\to\infty} \pi(x)/(x/\log x) = 1$).

Considering *the Prime Number Theorem*, we conclude that about $(\log 10^{100})/2 = 115$ numbers will be tested before a prime is found.

To test whether the random number is prime or not, the *Solovay* and *Strassen* method can be used. It picks a random number a from a uniform distribution on $1, \ldots, b-1$, and tests whether the following statements are true:

$$\gcd(a, b) = 1$$

and

$$J(a, b) = a^{(b-1)/2} \pmod{b}$$

Finding *Jacobi* and *Greatest Common Divisor* of two numbers can be found efficiently.

3 RSA Security and Potential Attacks

To break a RSA encrypted code, one approachis to find the private-key. To find the private-key, either $\phi(n)$ should be known, or d should be guessed.

If d has been chosen wisely from a large set, guessing d is not easier than factoring n, to find $\phi(n)$. Factoring algorithms are not efficient enough for large numbers. (i.e. Polards algorithm factors a number n in time $O(n^{1/4})$), therefore security of RSA is based on difficulty of factoring, good choice of public and private keys and frequency of usage over the network.

3.1 Elementary Attacks on RSA Protocol

Misusing RSA over the network gives the opportunity for eavesdropping. Here are two cases of attacks against RSA protocol:

3.1.1 Common Modulus

This is the case that all users, although using different public and private keys, are using a common n.

$$n = e_{alice}d_{alice} = e_{bob}d_{bob} = \cdots$$

Cryptanalysis:[5]

$$\left.\begin{array}{ll} e_{bob} & \text{is known by Eve} \\ e_{alice} & \text{is known by Eve} \\ \exists r, s| & re_{bob} + se_{alice} = 1 \end{array}\right\} \Rightarrow r \text{ and } s \text{ are computable by use of Euclidian algorithm}$$

Also:

$$\begin{aligned} C_{alice} &\equiv M^{e_{alice}} \pmod{n} \\ C_{bob} &\equiv M^{e_{bob}} \pmod{n} \end{aligned}$$

Therefore the original message can be computed :

$$M \equiv (C_{alice}^{-1})^{-r} * C_{bob}^{s} \pmod{n}$$

3.1.2 Blinding

In blinding, Eve's goal is to figure out M from it's cyphertext C. However she knows that if she sends a request to Alice and ask her to decrypt C for her the request will be refused, since Alice knows that C is the decrypted form of M.

Therefore, Eve chooses a random number r and encrypts it with the available public key:

$$X \equiv r^{e} \pmod{n}$$

And also computes Y:

$$Y \equiv XC \pmod{n}$$

Now, Eve sends Y to Alice. This time Alice by decrypting Y will come up with garbage U. However Eve will be able to rebuild M from U.

$$\begin{aligned} r^{-1}U \pmod{n} &= r^{-1}Y^{d} \pmod{n} \\ &= r^{-1}X^{d}C^{d} \pmod{n} \\ &= C^{d} \pmod{n} \\ &= M \end{aligned}$$

3.2 Implementation Attacks on RSA

3.2.1 Timing Attacks

To break a smartcard[2] encoding, that uses RSA key, Eve sends m messages to the smartcard for decryption, then from the time it takes to decypher a message, Eve can rebuild private-key d. Note that smartcards use *repeated squaring algorithm* to decrypt a message.

Assume in binary form, $d = d_k d_{k-1} \cdots d_0$.

Here is how a repeated squaring algorithm works[2]:

[2]A smart card is a plastic card embedded with a computer chip that stores and transacts data between users.

1. $j \leftarrow 0$
2. $M \leftarrow 1$
3. let $\langle d_k d_{k-1} \cdots d_0 \rangle$ be a binary representation of d
4. **for** $i \leftarrow k$ **downto** 0
5. **do** $j \leftarrow 2j$
6. $M \leftarrow M \times M \pmod{n}$
7. **if** $d_i = 1$
8. **then** $j \leftarrow j + 1$
9. $M \leftarrow M \times C \pmod{n}$
10. **return** M

Eve knows that $d_0 = 1$, she can also guess from the time spent by the smartcard to iterate the algorithm above, if other digits are either 0 or 1.

3.2.2 Random Faults

Sometimes to speed up the computation of the decrypting process, Bob instead of computing C directly by applying private key d to the cypher text (i.e $C \equiv M^d \pmod{n}$) uses d_p and d_q where $n = pq$:

$$\begin{aligned} d_p &\equiv d \pmod{p-1} \\ d_q &\equiv d \pmod{q-1} \end{aligned}$$

In this case C can be computed faster by using multi-processors with less iteration[3]:

$$\left.\begin{aligned} M_p &\equiv C^{d_p} \pmod{p} \\ M_q &\equiv C^{d_q} \pmod{q} \end{aligned}\right\} \Rightarrow M \equiv T_p M_p + T_q M_q \pmod{n}$$

Now the problem occurs when because of a hardware problem M does not compute properly for example:

$$\hat{M} = T_p M_p + T_q \hat{M}_q$$

Eve by finding $\hat{M}$ can efficiently factor n since, $\gcd(n, \hat{M} - C)$ expresses a non-trivial factor of n.

---

[3] $T_p \equiv 1 \pmod{p}$
$T_p \equiv 0 \pmod{q}$

3.3 Low Private Decryption Exponent

M.Wiener shows the small private hey results in a total break of the cryptosystem. In fact, d[5] can be recovered when:

$$\begin{array}{l} d < n/4 \\ \& \\ e < n \end{array}$$

Theorem 3.1. *[1].*
Let $n = pq$ with $q < p < 2p$. Let $d < \frac{1}{3}n^{1/4}$. Given $\langle n, e\rangle$ with $ed \equiv \pmod{\phi_{(n)}}$. d can efficiently be recovered.

$$\begin{array}{ll} \text{If:} & ed - k\phi_{(n)} = 1 \\ \text{then} & |\frac{e}{n} - \frac{k}{d}| \leq \frac{1}{dn^{1/4}} < \frac{1}{2d^2} \quad \text{which is a classic approximation relation} \end{array}$$

3.4 Low Public Encryption Exponent

3.4.1 Hastad's Broadcast Attack

As an example, if the encryption exponent 3 is used for the RSA cryptosystem by three different people with different moduli, a plaintext message M encrypted using each of their keys can be recovered from these resulting three ciphertext messages[4].

$$\left.\begin{array}{ll} C_1 \equiv M^3 & (\text{mod } n_1) \\ C_2 \equiv M^3 & (\text{mod } n_2) \\ C_3 \equiv M^3 & (\text{mod } n_3) \end{array}\right\} \Rightarrow C' \equiv M^3 \pmod{n_1 n_2 n_3}$$

where, $\gcd(n_i, n_j) = 1$ if $i \neq j$.
We know, $M < n_i$, therefore, $M^3 < n_1 n_2 n_3$.
From this information, Eve could recover M by computing the real cube root of C'.

The easiest solution to prevent this kind of attack is to pad messages with independent random values[5].

3.4.2 Partial Key Exposure Attack

Boneh, Durfee, and Frankel [1] showed that as long as $e < \sqrt{n}$ and Eve knows a fraction of the bits of d, she can reconstruct the rest of the bits of d.

$$\begin{array}{ll} \text{If} & ed - k(n - p - q + 1) = 1 \\ \text{while} & \hat{d} = \lfloor (kn + 1)/e \rfloor \\ \text{then} & |\hat{d} - d| \leq k(p + q)/e \leq 3k\sqrt{n}/e < 3\sqrt{n} \end{array}$$

Hence, $\hat{d}$ is a good approximation for d.

4 The Rabin Crytosystem

The Rabin cryptosystem is computationally secure against a chosen-plaintext attack provided that the modulus $n = pq$ cannot be factored [6].

Let n be the product of two distinct primes p and q, where:

$$p \equiv 3 \pmod 4$$
$$q \equiv 3 \pmod 4$$

then:

$$e(x) \equiv x(x+b) \pmod n$$
$$d(y) \equiv \sqrt{\frac{B^2}{4} + y} - \frac{B}{2} \pmod n$$

where, $0 < B < n - 1$.

References

[1] Dan Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the American Mathematical Society (AMS)*, 46(2):203–213, 1999.

[2] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press/McGraw-Hill, 1990.

[3] R. L. Rivest, A. Shamir, and L. M. Adelman. A method for obtaining digital signatures and public-key cryptostems. Technical Report MIT/LCS/TM-82, 1977.

[4] Kenneth H. Rosen. *Elementary Number Theory*. Addison Wesley Longman, 2000.

[5] Bruce Schneier. *Applied Cryptography Protocols, Algorithms, and Source Code in C*. John Wiley and Sons, Inc., 1994.

[6] Douglas R. Stinson. *Cryptography Theory and Practice*. CRC Press,Inc., 1995.