RSA Public Key system: Modern Cryptanalysis

Senior Research from 6/5/17 - 8/18/17

Juan G. Vargas Jr.

Contents

Abstract

In today's world, electronic communication helps our society maintain it's futuristic structure.With our construction of "big data", systems have the utmost important task of protecting the integrity of that data. This project dives into one of the most widely used public-key system, the RSA cryptosystem.

1 Components of a Cryptosystem

A Cryptosystem is the actual implementation of cryptographic techniques and infrastructure. In a public key cryptosystem the process requires two different keys; one used for encryption and the other for decryption. Another important property of a public key scheme is the guarantee that each recipient has a unique private key used for decryption. Even though there are different schemes, such as symmetric, asymmetric, and hybrid (a combination of both symmetric and asymmetric); all practical system will still consist of these components.

- **Plaintext**, this is the data that is transfered between nodes.
- **Ciphertext**, this is the encrypted version of the plaintext data file. The ciphertext is produced by feeding in the plaintext into an encryption algorithm.
- **Encryption algorithm**, the mathematical process that takes a plaintext and an encryption (public) key and produces a ciphertext
- **Decryption Algorithm**, also uses some mathematical process that produces a unique plaintext for any given ciphertext and decryption (private) key.

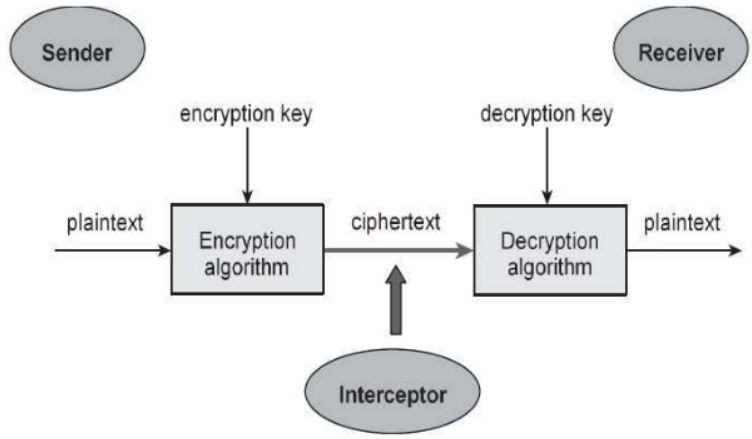

Figure 1: This is a basic cryptosystem design

A common philosophy cryptosystems follow is Kerckhoff's principal: a cryptosystem should be secure even if the attacker knows all the details about the system, with the exception of the secret key. In particular, the system should be secure when the attacker knows the encryption and decryption algorithms.

1.1 Cryptosystem Security Services

A modern cryptosystem provides four primary services.

- **Confidentiality**, keeps information from an unauthorized users. Confidentiality is achieved with either physical securing or mathematical algorithms for encryption.

  30

- **Data Integrity**, is a service that identifies any alteration to the data. Data may be modified by unauthorized entity intentionally or accidentally. Data integrity confirms whether data is intact or manipulated since it was last created, transmitted, or stored by an authorized user. **Note:** Data integrity does not prevent the alteration of data, but states whether data has been manipulated.

- **Authentication** relays the identification of the originator of a data packet. Confirms to the receiver that the data received has been sent only by an identified and verified sender.

  - **Message Authentication** identifies the originator of the message without any regard router or system that has sent the message
  - **Entity Authentication** is assurance that data has been received from a specific entity, say a particular website.

- **Non-repudiation** ensures that an entity cannot refuse the ownership of a previous commitment or an action. Assures that the original creator of the data cannot deny the creation or transmission of the said data to a recipient or third party. This is used for dealing with situations like an online transaction. Once an order is placed electronically, a purchaser cannot deny the purchase order if Non-repudiation service was used,

1.2 Public-Key Infrastructure

The big advantage in a Public-Key scheme is the ability to freely distribute public keys, as opposed to symmetric key distribution but creates problems of its own. If a public key is easily available in a public domain it is prone to interception in order to gain access to a system. In response to such an attack there must be some infrastructure to establish a trusted channel, like the security authenticators mentioned above, and some sort of key manager.

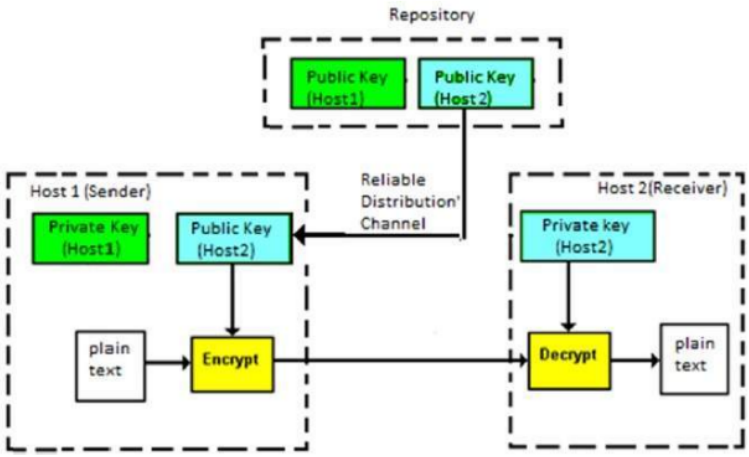

Figure 2: This is a basic Asymmetric-key cryptosystem scheme

2 Essential RSA Functions

The RSA scheme relies on the difficulty of factoring a very large number in order to find out the factors used in constructing variable 'n' and methods to easily simplify very large exponent computations, or exponentiation. These theorems are used to maintain the structure of the mathematical principals around the RSA scheme.

2.1 Number Theory Components

A common requirement for the RSA infrastructure is certain variables must be **Relatively prime**. Numbers are relatively prime when they have no common factors other than 1, in other words you cannot evenly divide both by some common value other than 1.

Theorem 1. ***Fermat's Little Theorem***
Let a be an integer and p be a prime, then:

$$a^{p-1} \equiv 1(modp)$$

A generalization of Fermat's Little theorem to any integer moduli is Euler's theorem.

Theorem 2. ***Euler's Theorem***
Let a and n be integers with gcd(a,n) = 1, then:

$$a^{\phi(n)} \equiv 1(modn)$$

Extended Euclidean Algorithm (EEA)
Input: positive integers r_0 and r_1 with $r_0 > r_1$
Output: gcd(r_0, r_1), as well as s and t such that gcd(r_0, r_1) = $s \cdot r_0 + t \cdot r_1$
Initialization:
$s_0 = 1$ $t_0 = 0$
$s_1 = 0$ $t_1 = 1$
i = 1
Algorithm:
DO
    i = i + 1
    $r_i = r_{i-2}$
    $q_{i-1} = (r_{i-2} - r_i) / r_{i-1}$
    $s_i = s_{i-2}$
    $t_i = t_{i-2} - q_{i-1} \cdot t_{i-1}$
WHILE
RETURN
    gcd(r_0, r_1) = r_{i-1}
    $s = s_{i-1}$
    $t = t_{i-1}$

2.2 Fast Exponentiation

Square-and-Multiply for Modular Exponentiation
Input:
base element 'x'
exponent 'e'
Output:
returns x · y where y is the result of binary exponentiation
Pseudo Code:

```
if e is zero
    y = 1
while e is greater than 1
    if e is even then
        x = x · x
        e = e/2
    else          y = x · y
        x = x · x
        e = (e -1)/2
return x · y
```

2.3 Optimization Techniques

3 The RSA Cryptosystem

3.1 Key Generation

There are two keys that must be generated; the public and private key. RSA Key Generation is a critical part of the RSA cryptosystem because the values chosen at this level determines how secure the algorithm is. For example, when selecting encryption value 'e' there must be a unique inverse so any value will not work. Values chosen at this point must to checked in order to ensure that the algorithm is in a sable form.

RSA Key generation
Output: public key = k_{pub} = (n,e) and private key =

1. Choose two large prime numbers p and q
2. Compute n = p · q
3. Compute Φ(n) = (p - 1)(q - 1)
4. Select the public exponent e ∈ 1,2, such that
$$gcd(e, \Phi(n)) = 1$$
5. Compute the private key d such that
$$d \cdot e \equiv 1 mod \Phi(n)$$

Note: The condition that gcd(e, Φ(n)) = 1 ensures that the inverse of e exists modulo Φ(n), so that there is always a private key d.

In accordance with Step 1, prime number selection for 'p' and 'q' should be about 512 bits in size. This is because the RSA modulus value n (computed in Step 2) should be at least 1024 bits in order to meet practical standards. It is also worth noting that both these values, p and q, must be co-prime of each other. By the very nature of primes, they have no other factors other than themselves and the number one so this should not be a concern unless number selection algorithm does not guarantee that the numbers chosen are prime.

3.2 Encryption and Decryption

3.3 RSA Proof of Correctness

In order to define RSA as a valid cryptosystem algorithm, we prove it's generality for any message x that we wish to encrypt and decrypt.

Theorem 3. *For all integers x, when e > 0*

$$x^{ed} \equiv x(mod\, n)$$

Suppose that gcd(x,n) = 1 and start with the construction rule for public and private keys:

$$e \cdot d \equiv 1(mod \phi(n))$$

Using an alternative representation of the modulo operator yields:

$$e \cdot d \equiv 1 + t \cdot \phi(n)$$

Coefficient t is found using the EEA when gcd($\phi(n)$, e) = s ·ϕ(n) + t ·e Then comes the application of this equivalence relation. We also can assume that when e·d is applied as an exponent to x the outcome is x^1 since encryption and decryption leaves the original message x.

$$x^{ed} \equiv x^{1+\phi(n)t} \equiv x^1 \cdot (x^{\phi(n)})^t \equiv (x^{\phi(n)})^t \cdot x(mod\, n)$$

Recalling Euler's Theorem 2 we can manipulate the equation due to the modulo's symmetric property to $1 \equiv \phi(n)\, mod\, n$. We now can make a generalization:

$$(\phi(n))^t mod\, n \equiv (1)^t \equiv 1$$

With this generalization we now have:

$$(x^{\phi(n)})^t \cdot x \equiv 1 \cdot x(mod\, n)$$

Thus proving that:

$$(x^{\phi(n)})^t \equiv x(mod\, n)$$

3.4 RSA Example

Before any process can begin the key pairs must be generated. This process yield us with values for: *p, q, n, $\phi(n)$, and e.*

NOTE: In order to remain within a reasonable bit range, this example will merely encrypt the plaintext '4'.

The selection of prime numbers p and q must be relatively prime (as shown in RSA proof).

$$p = 3, q = 11$$

Yielding the following values for n and ϕ(n).

$$n = p * q = 33$$

$$\phi(n) = (p-1)(q-1) = 20$$

Next is the selection of encryption value 'e' and selection must ensure that the inverse of value 'e' exists (e value requirements are listed in 2.1) The inverse of e is given by the Extended Euclidean Algorithm and is equivalent to decryption value 'd'

$$e = 3, d = 7$$

This ends the key generation phase as all required values for the clients and system are defined.

4 Project Walk through

The goal of this analysis is to mimic the behavior of an RSA cryptosystem and simulate an Active Attack, when attacker changes the information being sent in some way. This will be achieved either by setting up a virtual machine instance and seek to obtain information sent from their PC or setting up a program to dynamically illustrate the relationship between security and

5 Timeline

June 5th: Start

Week 1: Strongly identify an area of cryptology to demonstrate and contribute to.

Week 2: Understand the topic and demonstrate the ability to explain outlined concepts to others.

Week 3: Learn and setup appropriate environments in order to continue towards intercepting data between two nodes.

July 18th: Prove the correctiveness of the RSA encryption and decryption mathematical equations

August 18th: End

5.1 How to add Citations and a References List

You can upload a `.bib` file containing your BibTeX entries, created with JabRef; or import your Mendeley, CiteULike or Zotero library as a `.bib` file. You can then cite entries from it, like this: [**?**]. Just remember to specify a bibliography style, as well as the filename of the `.bib`.

You can find a video tutorial here to learn more about BibTeX.

We hope you find Overleaf useful, and please let us know if you have any feedback using the help menu above — or use the contact form at `https://www.overleaf.com/contact`!

References