College of Computer & Information Science — CS 1800
Northeastern University — Discrete Structures

The RSA (Rivest-Shamir-Adleman) cryptosystem

This material is outside the scope of this course, so you are not expected to master these contents. We do hope that it piques your interest enough to read and explore more.

1 Introduction

The RSA cryptosystem is a public-key cryptosystem, widely used for secure communication and e-commerce applications. It is often used to encrypt messages sent between two communicating parties so that an eavesdropper who overhears the conversation cannot decode them easily. It also enables a party to append an unforgeable signature to the end of a message. This signature cannot be "easily" forged and can be checked by anyone.

2 How do public-key cryptosystems work?

Consider our protagonists Alice and Bob who want to communicate with each other securely. Suppose Bob wants to send a message to Alice. In a typical public-key cryptosystem Alice has two keys, a secret (or private) key that only Alice knows and a public key that Alice advertises to the whole world. Each key yields a function that map a message to another message: the public key yields a public *encryption function* – let us call it P_A – and the secret key yields a secret *decryption function* S_A. A typical message exchange proceeds as follows.

Bob encrypts his message M using P_A and sends the message $P_A(M)$ to Alice.

If Alice receives $P_A(M)$, she applies S_A and obtains $S_A(P_A(M)) = M$.

So we want two functions S_A and P_A such that $S_A(P_A(M)) = M$, for all permissible messages M. Furthermore, our selection of S_A and P_A should be such that any eavesdropper, who has P_A and can read message $P_A(M)$, cannot "efficiently" extract M from this; or, ideally, cannot "efficiently" extract any reasonable information from this.

3 How does RSA work?

RSA, due to Ronald Rivest, Adi Shamir, and Leonard Adleman, is the most popular public key cryptosystem today, widely used in commercial applications. The basic RSA cryptosystem is completely specified by the following sequence of steps.

1. Alice selects at random two large primes p and q.

2. Alice computes $n = pq$.
3. Alice selects a small odd integer e that is relatively prime to $(p-1)(q-1)$.
4. Alice sets d so that $de \bmod (p-1)(q-1)$ equals 1.
5. Alice publish the pair (e, n) as the public key, with $P_A(M) = M^e \bmod n$.
6. Alice stores the pair (d, n) as the secret key, with $S_A(E) = E^d \bmod n$.

In order to send message M in $\{0, 1, \ldots, n-1\}$, Bob sends $P_A(M) = M^e \bmod n$. On receiving the encrypted message Alice computes $S_A(P_A(M)) = M^{de} \bmod n$. Our choices of d, e, and n ensure that $M^{de} \bmod n$ equals M.

To see the last step, we need to understand the mathematical underpinnings of RSA, which we will do so in Section 4. But first, let us do a couple of examples. Take

$$p = 5, q = 3; n = 5 \cdot 3 = 15; (p-1)(q-1) = 4 \cdot 2 = 8; e = 3.$$

We need to find the multiplicative inverse of 3 mod 8. Well, $3 \cdot 3 = 9 \bmod 8 = 1$. So $d = 3$.

Here is another example. Suppose we take

$$p = 7, q = 11; n = 7 \cdot 11 = 77; (p-1)(q-1) = 6 \cdot 10 = 60; e = 13.$$

What is d? Note that d is the multiplicative inverse of $e \bmod 60$. We can calculate it using the Extended Euclid algorithm to obtain 37; you can verify that $37 \cdot 13 \bmod 60 = 1$.

Suppose the message $M = 2$. Then the encrypted message is $2^{13} \bmod 77$. We calculate it as follows:

$$\begin{aligned}
2^2 \bmod 77 &= 4 \\
2^4 \bmod 77 &= 16 \\
2^8 \bmod 77 &= 256 \bmod 77 = 25 \\
2^{13} \bmod 77 &= (25 \cdot 16 \cdot 2) \bmod 77 = 800 \bmod 77 = 30.
\end{aligned}$$

To decrypt, Alice computes $30^{37} \bmod 77$, which can be calculated as follows.

$$\begin{aligned}
30^2 \bmod 77 &= 900 \bmod 77 = 53 \\
30^4 \bmod 77 &= 53^2 \bmod 77 = 2809 \bmod 77 = 37 \\
30^8 \bmod 77 &= 37^2 \bmod 77 = 1369 \bmod 77 = 60 \\
30^{16} \bmod 77 &= 60^2 \bmod 77 = 3600 \bmod 77 = 58 \\
30^{32} \bmod 77 &= 58^2 \bmod 77 = 3364 \bmod 77 = 53 \\
30^{37} \bmod 77 &= (30^{32} \cdot 30^4 \cdot 30) \bmod 77 = (53 \cdot 37 \cdot 30) \bmod 77 = 2.
\end{aligned}$$

4 Why does RSA work?

We can prove that RSA works by showing that for any M in $\{0, \ldots, n-1\}$, the following equation holds.

$$M^{de} \bmod n = M \tag{1}$$

We establish the above Equation using a nice theorem due to the French mathematician Pierre Fermat, called Fermat's Little Theorem.

Theorem 1 *Fermat's Little Theorem If p is prime, then for all $1 \leq a < p$, we have*

$$a^{p-1} \bmod p = 1.$$

Proof: Consider $a \cdot x$ and $a \cdot y$ for $x \neq y$, $1 \leq x, y < p$. We claim they are different mod$p$ since otherwise, p divides a or $x - y$, both not possible. So, $a \cdot 1 \bmod p$, $a \cdot 2 \bmod p$, $\ldots$, $a \cdot (p-1) \bmod p$ are all different numbers in $\{1, 2, ..., p-1\}$. Thus, we have

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) = a^{p-1} \cdot (p-1)! = (p-1)! \bmod p.$$

Thus, p either divides $a^{p-1} - 1$ or divides $(p-1)!$. The latter is not possible, hence the claim. ■
We now establish Equation 1. Recall that $de \bmod (p-1)(q-1) = 1$. So $de = k(p-1)(q-1) + 1$ for some integer k. We will show that $M^{(p-1)(q-1)} \bmod n$ is equal to 1. Note that this immediately implies that $M^{de} \bmod n = M$.

We consider two cases. In the first case, M is relatively prime to both p and q; in the second, M has a common factor with either p or q.

1. Let us consider the first case. By Fermat's Little Theorem, we know that $M^{p-1} \bmod p = 1$ and $M^{q-1} \bmod q = 1$. Therefore, we have $M^{(p-1)(q-1)} \bmod p$ and $M^{(p-1)(q-1)} \bmod q$ are both 1. Thus, $M^{(p-1)(q-1)}$ is of the form $k_1 p + 1$ as well as of the form $k_2 q + 1$ for some integers k_1 and k_2. This implies that $k_1 p = k_2 q$; since p and q are different primes, this can hold only if k_1 is a multiple of q and k_2 is a multiple of p. It thus follows that $M^{(p-1)(q-1)}$ is of the form $kpq + 1 = kn + 1$ for some integer k. In other words, $M^{(p-1)(q-1)} \bmod n$ equals 1. This completes the proof for the first case.

2. We now consider the case where M shares a common factor with either p and q. In this case, M is a multiple of p or q. Suppose, without loss of generality $M = kp$ for some integer k. Let us consider what $M^{de} \bmod p$ and $M^{de} \bmod q$ are. Since M is a multiple of p, $M^{de} \bmod p = 0$. So M^{de} is a multiple of p. We now calculate $M^{de} \bmod q$ as follows. Since $M < n$, it follows that $k < q$; so M is relatively prime to q. By Fermat's Little Theorem, we then have $M^{q-1} \bmod q = 1$. Therefore, we also have $M^{(p-1)(q-1)} \bmod q = 1$, implying that $M^{de} \bmod q = M \bmod q$. We thus have

   $$M^{de} \bmod p = 0; M^{de} \bmod q = M \bmod q.$$

   Let M_1 denote $M^{de} \bmod pq$. It follows that

   $$M_1 \bmod p = 0; M_1 \bmod q = M \bmod q.$$

   Thus M_1 is of the form ℓp for some integer $\ell < q$. But since q divides $M - M_1 = (k - \ell)p$ for nonnegative integers $k, \ell < q$, it follows that $k = \ell$, implying that $M_1 = M$.

We have thus shown that the decryption function of RSA, when applied to an encrypted message, yields the original unencrypted message as desired.

5 Is RSA secure and efficient?

What are the individual steps in RSA? Which of these can be executed efficiently? How secure is RSA? Here is a brief discussion on the efficiency and security of RSA.

- Generating two large random primes: How do we perform this? Well, one way to do it is to generate a random number and then test whether it is prime. How do we test whether it is prime? Our naive scheme (that works in time proportional to the square root of the number) is too slow and inefficient. Fortunately, there are faster, efficient, ways to do it.
- We need to raise a number to a (potentially) large power in modular arithmetic. We have seen how to do this efficiently using the repeated squaring method.
- We also need to find a multiplicative inverse in modular arithmetic. This can be done efficiently using the Extended Euclid Algorithm.
- We do not want the number n (used in the RSA private and public key) to be easily factored into its prime factors p and q. A naive algorithm takes time proportional to the square root of n. Fortunately, there is *no* efficient way known for this problem.
- As stated, RSA is a deterministic encryption system; i.e., a particular message is encrypted the same way every time. This is prone to easy attacks, referred to as plaintext attacks, where the attacker may be aware that the message being sent is one of a small number of possibilities, and can try the public encryption system with different possibilities. One way to avoid this attack, is making the system randomized – for instance by adding a fixed-size random pad to the plaintext message and encrypting the padded message. The random pad is chosen independently at every step, thus making the above plaintext attack more difficult.