THE RSA CRYPTOSYSTEM

SILVIA ROBLES

Abstract. This paper explores the history and mathematics behind the RSA cryptosystem, including the idea of public key cryptosystems and number theory. It outlines the applications of RSA including secure information transfers and electronic signatures. It also analyzes why RSA is a secure way of transmitting information and the ways that it has been defeated in the past. In particular, we focus on the method of the quadratic sieve, a version of which was used to crack RSA-129.

1. Introduction

In 1977 the internet, electronic mail, and electronic banking were in their infancy. Many of the technological advances that we take for granted today were just beginning to be developed and refined. If we put ourselves in the shoes of the many scientists who were working on designing the internet what are some problems that we would encounter? One of the major obstacles, especially for email and banking, was security and privacy. There had to be some way to send messages across phone lines without unwanted eavesdroppers being able to intercept and understand them. The integrity of sensitive information such as social security numbers and bank accounts depended on it. Thus the science of cryptography became incredibly important for the success of the internet.

The RSA cryptosystem was first proposed in 1977 by Ronald Rivest, Adi Shamir, and Len Adleman [6]. It is from the initials of their last names that the system derived its name. Prior to RSA people would encipher their messages and send a courier to the recipient of the message with the key to decipher it. The problem with this, of course, was making sure that the message the courier carried was secure. RSA was novel in that making the encryption key public did not reveal the decryption key [6]. It is a shining example of Public Key Cryptography, a concept first proposed by Diffie and Hellman in their classic 1976 paper[4]. Today, RSA is widely used and is regarded as one of the most secure cryptosystems in existence.

Date: May 9, 2006.

2. Public Key Cryptosystems

Let M be the set of all possible messages and K be the set of all "keys." For each key $k \in K$ there exists both a decryption function $D_k : M \to M$ and an encryption function $E_k : M \to M$. In order to be considered a public key cryptosystem these functions must satisfy the following conditions [7]:

(1) For every $m \in M$ and every $k \in K$, $E_k(D_k(m)) = m$ and $D_k(E_k(m)) = m$.
(2) For every $m \in M$ and every $k \in K$, the values of $E_k(m)$ and $D_k(m)$ are not difficult to compute.
(3) For almost every $k \in K$ if somebody knows only the function E_k, it is computationally infeasible to compute D_k.
(4) Given $k \in K$, it is easy to find the functions E_k and D_k.

A function E_k that satisfies (1)-(4) is called a *trap-door one-way permutation* [6]. The function is *one-way* because it is easy to compute in one direction but not in the other. The *trap-door* refers to the fact that the inverse functions become simple to compute once certain information is revealed, in other words, once one finds the trap-door. It is a *permutation* because every message is an encryption of another message and every encrypted message is also a permissible message. This property is useful when trying to find a way to "sign" electronic documents.

Consider two people who are trying to communicate a private message, José and Silvia. Let their encryption and decryption functions be denoted as D_J, E_J and D_S, E_S. They both put their encryption functions E_J and E_S in their public files. José sends a message to Silvia by recovering E_S from her public file and encrypting his message with it. He sends Silvia $E_S(m)$ and only she can decipher it since only she knows D_S. Likewise, she can send a message m' to José by using E_J.

3. RSA

The basic premise behind the RSA cryptosystem is that although multiplying two numbers is a simple process, factoring the product back into the original two numbers is much more difficult to do computationally. The difficulty increases as we use larger and larger numbers. In order to encrypt a message using the RSA cryptosystem one must first choose two large prime numbers p and q, usually of about 50 digits each.

Definition 3.1. An integer p is *prime* if $p \neq \pm 1$ and it's only divisors are ± 1 and $\pm p$. An integer that does not satisfy the previous property is called *composite*.

After choosing p and q we get $n = p * q$. The encryption key is the pair of integers (e, n) and the decryption key is the pair (d, n). Given a message m, in order to encrypt it we would first represent it as an integer between 0 and $n - 1$. If the message is too large then we can break it into blocks, as long as each block is between 0 and $n - 1$. Then encrypt m by raising it to the e^{th} power modulo n. Denote the resulting ciphertext as c. Then we have

$$E(m) \equiv m^e \equiv c \pmod{n}.$$

To decrypt the ciphertext we would raise it to the d^{th} power modulo n.

$$D(c) \equiv c^d \equiv m \pmod{n}.$$

The integers e and d are closely related to p and q. Choose d to be any large random integer that is *relatively prime* to $(p-1)(q-1)$. Then e is the *multiplicative inverse* of d, modulo $(p-1)(q-1)$.

Definition 3.2. The integers a and b are *relatively prime* if their greatest common divisor is 1,

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

Definition 3.3. An integer a is the *multiplicative inverse* of an integer b, modulo s, if

$$a * b \equiv 1 \pmod{s}.$$

Anybody wishing to send and receive private messages would make their encryption key (e, n) public but keep their decryption key (d, n) private. Nobody would be able to derive the private key from the public key due to the extreme difficulty of factoring n.

We can now prove that the above method fulfills the properties of a public key cryptosystem.

Definition 3.4. Let $\varphi(n)$ be the *totient function* of an integer n. Then $\varphi(n)$ gives the number of positive integers that are relatively prime to n. That is, $\varphi(n)$ gives every integer $a > 0$ that satisfies $gcd(a, n) = 1$.

Theorem 1. *The RSA cryptosystem fulfills property (1) of a public key cryptosystem.*

Proof. By Euler's theorem [3] we know that for any two integers m and n that are relatively prime

$$m^{\varphi(n)} \equiv 1 \pmod{n}. \tag{1}$$

Also, for a positive prime integer p,

$$\varphi(p) = p - 1.$$

Therefore, for any message m that is relatively prime to $n = p * q$,

$$m^{\varphi(n)} \equiv 1 \pmod{n}$$

By the basic properties of the totient function $\varphi(n)$ [3] we know

$$\begin{aligned} \varphi(n) &= \varphi(p) * \varphi(q), \\ &= (p-1)(q-1) \\ &= n - (p+q) + 1. \end{aligned}$$

Also, we have defined the encryption and decryption functions in such a way that $E(D(m)) \equiv D(E(m))$ modulo n since

$$\begin{aligned} E(D(m)) &\equiv (D(m))^e \equiv (m^d)^e \pmod{n} \\ D(E(m)) &\equiv (E(m))^d \equiv (m^e)^d \pmod{n}. \end{aligned}$$

Furthermore, by construction we have chosen e and d to be multiplicative inverses modulo $(p-1)(q-1)$. In other words we have

$$e * d = k * \varphi(n) + 1 \quad \text{for any integer } k. \tag{2}$$

By (1) we see that for all messages m such that p does not divide m

$$m^{(p-1)} \equiv 1 \pmod{p}.$$

By (2) and since $(p-1)$ divides $\varphi(n)$ we have

$$\begin{aligned} m^{k*\varphi(n)+1} &= m * (m^{(p-1)})^{k(q-1)} \\ &\equiv m \pmod{p}. \end{aligned}$$

The above equality holds for all m. If we use a similar argument for q we see, for all m,

$$m^{k*\varphi(n)+1} \equiv m \pmod{q}.$$

If we combine the two previous equations we deduce

$$m^{e*d} \equiv m^{k*\varphi(n)+1} \equiv m \pmod{n}$$

for all m. □

Example 3.1. Let $p = 47$, $q = 59$, $n = 47 * 59 = 2773$, and $d = 157$. Then $\varphi(2773) = 46 * 58 = 2668$. The Extended Euclidian algorithm is a procedure to find α, β, and c where $\alpha * a + \beta * b = c$, for integers a, b, and c such that $gcd(a, b) = c$ [3]. If we compute e using the Extended Euclidean algorithm we can set $a = \varphi(n)$, $b = d$ and we know that since $\varphi(n)$ and d are relatively prime at the end of the algorithm we will obtain $c = 1$. However, we will also obtain α and β where $\alpha * \varphi(n) + \beta * d = 1$, and β will be the multiplicative inverse of $d \pmod{\varphi(n)}$. Using this procedure we arrive at $e = 17$. Suppose we have a message

$$m = \text{IT'S ALL GREEK TO ME.}$$

Then we can set each letter in the alphabet equal to a two-digit number. This will ensure that there is no ambiguity when encoding and decoding. If $A = 1$, $B = 2$,..., then 12 could mean AB or L. Therefore we set $blank = 00$, $A = 01$, $B = 02$, ..., $Z = 26$. The encoded message becomes

$$m = 0920\ 1900\ 0112\ 1200\ 0718\ 0505\ 1100\ 2015\ 0013\ 0500.$$

Note that the message has is broken into blocks of two letters each. If we put it in blocks of three letters they would not each be less than $n - 1 = 2772$. Let m_1 be the first block of the message. Then to encipher m_1 we calculate

$$E(m_1) \equiv (m_1)^{17} \equiv (920)^{17} \equiv 948 \pmod{2773}.$$

Let c denote the ciphertext for the entire message, then

$$c = 0948\ 2342\ 1084\ 1444\ 2663\ 2390\ 0778\ 0774\ 0219\ 1655.$$

It is easy to check that the deciphering method works. For m_1, $948^{157} \equiv 920 \pmod{2773}$.

4. Signatures

The RSA cryptosystem can be used to send and receive sensitive or private information over insecure channels. However, it also has another use which has proven equally important. That is, it allows people to sign electronic documents. There are two obstacles to signing documents over the internet. The first, of course, is making sure that the signature is not forged by an impostor. The second, is verifying that the message that is signed is not altered in any way after it has

been signed. RSA provides a clever answer to both of these problems. It assures that a signature is paired with a certain message and cannot be "pasted" on to a different message. Let S denote the signature for the message m. Recall our two people, Silvia and José. Each had their corresponding encryption and decryption functions, E_S, D_S, and E_J, D_J respectively. If José is sending the message to Silvia he signs it by computing

$$S = D_J(m).$$

He then encrypts this signed message in the usual way, using E_S from Silvia's public file. The message that Silvia receives is

$$E_S(S),$$

which Silvia can decrypt to obtain S with her own private decryption function. Since Silvia is expecting a message from José she knows to extract the original message by using E_J from José's public file. Property (1) of public key cryptosystems ensures that $E_J(D_J(m)) = m$. The result is a message, signature pair (m, S). This process proves that José (and not some impostor) signed the message because only he could manufacture $S = D_J(m)$. If we change the message then the signature changes, therefore no signature can be "pasted" onto a different message.

5. Security of RSA

The security of the RSA cryptosystem is not perfect. There are several weaknesses that must be guarded against that mostly consist of avoiding prime numbers that are easily found by current factoring methods. This implies that the security of the RSA cryptosystem rests on the difficulty of factoring n. Indeed, trying to break RSA by finding d, the decryption key, or computing $\varphi(n)$, amounts to factoring n in the end.

Once we had $\varphi(n)$ we could break RSA in two ways. First, we could find d by finding the multiplicative inverse of $e \pmod{\varphi(n)}$. Second, we could find $n = pq$. To see why computing $\varphi(n)$ would give us p and q [7] note

$$\begin{aligned} n - \varphi(n) + 1 &= pq - (p-1)(q-1) + 1 \\ &= p + q. \end{aligned}$$

Once we know $n = p * q$ and $p + q$, we can find p and q by computing the roots of the polynomial

$$\begin{aligned} x^2 - (n - \varphi(n) + 1)x + n &= x^2 - (p+q)x + pq \\ &= (x-p)(x-q). \end{aligned}$$

Since finding $\varphi(n)$ allows us to factor n, it is no easier than factoring n. Similarly, since once d is known, n would be known, finding d can be no easier than factoring n.

To factor n once we have d we would first compute $de-1$. The term $de-1$ is a multiple of $\varphi(n)$. We can now apply the method for universal exponent to factor n since for any integer a such that $gcd(a, n) = 1$

$$a^{(de-1)} \equiv a^{k\varphi(n)} \equiv 1 \pmod{n}.$$

There are many factoring algorithms known today. Here are a few examples of how their existence would affect our choice of d, p, and q [7].

Theorem 2. *Let $n = pq$, where p and q are primes with $q < p < 2q$. Suppose $d < 1/3n^{1/4}$. Given (n, e) such that $d * e \equiv 1 \pmod{\varphi(n)}$ holds, there is an efficient procedure for computing d.*

Proof. The method is given by [8] and uses continued fractions for e/n. □

The result of the above theorem is that p and q should be of slightly different sizes and d should be large in order to guard against this particular attack.

Theorem 3. *Let $Sn = pq$ have t digits. If we know the first $t/4$, or the last $t/4$ digits of p then we can efficiently factor n.*

Proof. This result is given in [2]. □

The result here is that we should choose a p such that most of the digits are not predictable. If we choose our prime p by testing numbers for primality that are always of the form $N * 10^{50} + k$ for a random 50-digit number N and $k = 1, 3, 5, \ldots$, then an attacker will know 47 of the last 50 digits. They will all be zero!

Theorem 4. *Suppose (n,e) is an RSA public key and n has t digits. Let d be the decryption key. If we have at least the last $t/4$ digits of d then we can efficiently find d in time that is linear in $elog_2e$.*

Proof. This proof is given in [1]. □

Here we see that if the encryption key e is large then it is difficult to find d since the search is bounded as a function linear in $elog_2e$. This shows that we should not choose a small e.

6. The Quadratic Sieve

The Quadratic Sieve is a factoring algorithm that was invented in 1981 by Carl Pomerance [5]. It has been used with much success since then and a variant of it was used to solve RSA-129 in 1994 [7]. To factor n using the quadratic sieve we must first create a *factor base*.

Definition 6.1. A *factor base* is a finite set of small primes.

Next choose integers r that are close to $\sqrt{n}$. Calculate $Q \equiv r^2 \pmod{n}$. Try to factor Q using only the factor base we created earlier. By the unique factorization theorem [3] an integer $n \leq 2$ can be uniquely written in the form

$$n = p_1^{e1} p_2^{e2} \cdots p_k^{ek},$$

where $1 < p_1 < p_2 < \ldots < p_k$ are prime numbers and $e1, e2, \ldots, ek$ are positive integers. For example, if the factor base contains 2,3,5 and 7, then 600 can be written as $2^3 * 3^1 * 5^2 * 7^0$. If a list of factors has only even exponents then it is a perfect square. We are interested in perfect squares because they are solutions to the congruence $x^2 \equiv y^2 \pmod{n}$. Finding a square Q will lead us to a factor of n, which is why this algorithm is called the quadratic sieve.

If we don't immediately find a Q that is a perfect square we can combine several Q's in order to obtain one that is a square. For example, if one Q is almost square but it contains 7^1 as a factor, and another Q is almost square but it contains 7^3 as a factor, multiplying them will yield a perfect square and multiplying their corresponding r's will satisfy the quadratic congruence. We can see that the procedure for making a square combination of Q's is a linear algebra problem. We can solve this problem by constructing a matrix where each column corresponds to a prime in the factor base and each row is a Q. The entries of the matrix will be the exponents that correspond to each Q. Also, since all that matters is whether the exponent is odd or even, each entry can be represented as either a 1 for odd or 0 for even. So if j^{th} row corresponds to $Q = 600$ then the entries for that row would look like $\begin{array}{cccc} 1 & 1 & 0 & 0 \end{array}$. We combine two rows by adding them modulo 2. We can use Gaussian elimination to find a linear dependence, and if we keep track of the operations we performed we can recreate them on the actual Q and r values that the row in the matrix represents and obtain a factor of n. In order for this matrix to work we need at least as many congruences as the number of primes in the factor base.

The method is a sieve because to determine the Q and r values we set them up in an arithmetic progression and cross out values every so often, much like the sieve of Eratosthenes. The algorithm sums

the logarithms of the successful divisors, which gives greater weight to larger factors. Each Q for which the sum is larger than some threshold is likely to be "smooth", in particular, it can be factored quickly by trial division.

In 1994 Arjen Lenstra, Paul Leyland, Michael Graff, and Derek Atkins organized a massive effort to factor RSA-129. RSA-129 was used to encode a message set out in 1977 by the creators of RSA as a challenge to anybody who thought they could break the encryption. They offered \$100 to anybody who could do it before April 1st, 1982 [7]. The 129 referred to the number of digits in n. The algorithm that they used was called the multiple-polynomial quadratic sieve, where the quadratic congruence $Q \equiv r^2 \pmod{n}$ is replaced by polynomial relations [5]. Through the effort of 600 people, in 24 countries, on 1600 computers, RSA-129 was factored in 7 months. The factor base included all prime factors less than 16333610. It took 45 hours to perform Gaussian elimination on the matrix they made with 524,338 columns (corresponding to each prime factor) and 569,466 rows (corresponding to each congruence). The first three factorizations that were found were trivial, but the fourth gave the solution to RSA-129 [5]. What was the message that Rivest, Shamir and Adleman had encrypted in 1977?

THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE

7. Conclusion

Cracking an RSA encryption is certainly a formidable task. The fact that it has survived the test of time stands as a testament to it's security; 30 years in a world where technology is advancing at breakneck speed is nothing short of amazing. However, the death of RSA can be seen in the horizon. There are two ways in which it could be circumvented. Either a brilliant mathematician finds an efficient factoring algorithm, or computers speed up. The first possibility could happen tomorrow. As illustrated by the above example, methods improved enough between 1977 and 1994 to be able to factor RSA-129, although it took a great deal of effort. With respect to technology, either normal computers can improve, or quantum computers can become a reality, which would effectively make RSA obsolete. Either one of these options will most likely take some time. Even so computer scientists and mathematicians are already searching for different approaches to encrypting messages such as elliptic curve cryptosystems. What will be the standard 10 or 20 years from now? Only time can tell.

References

[1] D. Boneh. Twenty years of attacks on the rsa cryptosystem. *Amer. Math. Soc. Notices*, 46:203–213, 1999.
[2] D. Coppersmith. Small solutions to polynomial equations, and low exponent rsa vulnerabilities. *J. Cryptology*, 10:233–260, 1997.
[3] S.C. Coutinho. *The Mathematics of Ciphers.* A K Peters, Ltd., 1999.
[4] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, pages 644–654, Nov 1976.
[5] Brian Hayes. The magic words are squeamish ossifrage. *American Scientist*, 82, 1994.
[6] Adi Shamir Ronald Rivest and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, 21:120–126, 1978.
[7] Wade Trappe and Lawrence C. Washingtion. *Introduction to Cryptography with Coding Theory.* Prentice Hall, Inc., 2002.
[8] M. Wiener. Cryptanalysis of short rsa secret exponents. *IEEE Trans. Inform. Theory*, 36:553–558, 1990.