The RSA Algorithm:

A Mathematical History of the Ubiquitous Cryptological Algorithm

Maria D. Kelly

December 7, 2009

Abstract

The RSA algorithm, developed in 1977 by Rivest, Shamir, and Adlemen, is an algorithm for public-key cryptography. In public-key cryptography, users reveal a public encryption key so that other users in the system are able to send private messages to them, but each user has their own private decryption key. The key to ensuring privacy in a public-key cryptosystem is for it to be extremely difficult to derive the decryption key from the publicly available encryption key. The algorithm works by exploiting concepts from number theory, including Fermat's Little Theorem. In this paper, we explore the RSA algorithm, its definition, the underlying mathematics, and a number of attacks against the algorithm. Two vulnerabilities of the algorithm used with low public exponent are presented in detail. The paper concludes with an explanation of what the RSA algorithm will need to maintain security as technology continues to advance.

Introduction

Before the introduction of public-key cryptography by Diffie and Hellman [3], if two people wanted to communicate in private by encrypting messages sent between them, they would first need meet to agree upon methods for encoding and decoding the messages. The advent of the public-key cryptosystem made these kinds of meetings unnecessary because it allowed both parties to make their encryption procedures publicly available without compromising the privacy of their communication. A public-key cryptosystem is one in which each user places an encryption procedure E into a public file. Each user has a corresponding decryption procedure D, the details of which the user does not reveal to anyone

else. The key to ensuring the security of a public-key cryptosystem is for it to be extremely difficult to derive the decryption key from the publicly available encryption key. In order to qualify as a public-key cryptosystem, the encryption and decryption procedures must have the following properties [3] [7]:

1. Applying the decryption procedure to a message encrypted by the corresponding encryption procedure yields the original message. This can be expressed formally as,

$$D(E(M)) = M.$$

2. Both the encryption procedures and the decryption procedures are easy to compute.

3. Publicly revealing the encryption method E does not reveal any easy way to compute the corresponding decryption procedure D.

4. If a message M is first deciphered using the decryption procedure D and then the result is encrypted using the corresponding encryption procedure E, the final result gives the original message. Formally we write this as,

$$E(D(M)) = M.$$

Typically, an encryption (or decryption) procedure E consist of an **encryption key** and a **general method** for enciphering a message M using the key. The enciphered message is called the **ciphertext** C. In a public-key cryptosystem, everyone can use the same method for enciphering the message because the security of any given encryption procedure relies on the security of the decryption key. An encryption function which satisfies the first three properties given above is called a **trap-door one-way function**. Diffie and Hellman, who first introduced the concept, define a trap-door one-way function as function whose inverse, though it exists, is computationally infeasible to compute when given only the original function [3]. Though Diffie and Hellman were the first to introduce the idea of public-key cryptography and of trap-door one-way functions, the true emergence of public-key cryptography did not come until the introduction of the RSA algorithm.

The RSA Algorithm: A Realization of Public-Key Cryptography

The RSA algorithm, introduced in 1977 by Rivest, Shamir, and Adlemen, is an algorithm for public-key cryptography. RSA was the first and is still the most widely-used algorithm for public key cryptography and it is used for thousands of applications from e-mail encryption to secure online purchasing. It was the first cryptosystem to enable senders to "sign" each message they send so that the recipient has proof of who sent the message.

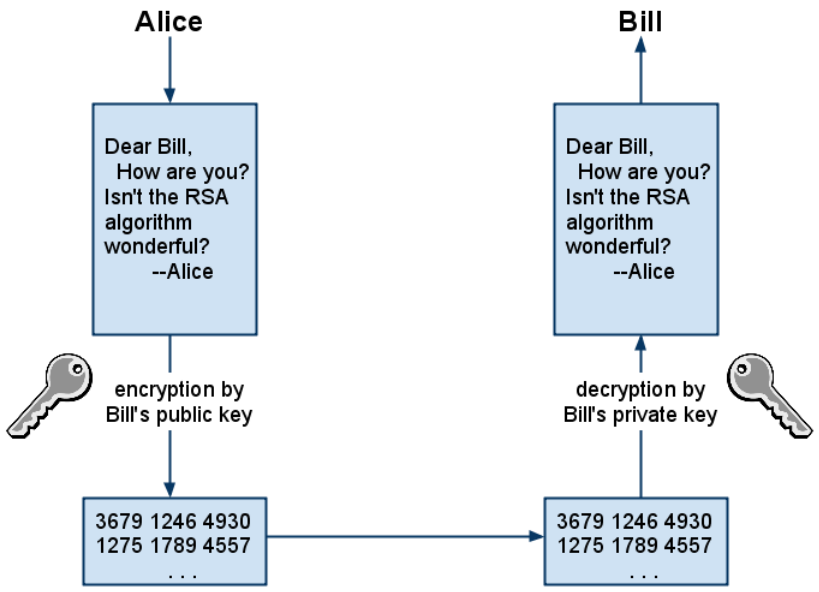

Figure 1: In public-key cryptography, if Alice wants to send a message to Bill a message, she first looks up Bill's public encryption procedure to encode the message, sends Bill the resulting ciphertext, and Bill is then able to decode the message using his private decryption procedure.

To encrypt a message using the RSA algorithm, given a public encryption key (e, n), the general method of the encryption procedure is as follows: The first step is to represent the message as an integer between 0 and $n-1$, M, using any standard representation. Then, to encrypt the message, raise M to the eth power modulo n. The ciphertext C is thus given by,

$$C \equiv E(M) \equiv M^e \pmod{n}.$$

To decrypt the message, we raise it to a different power, d, part of the private decryption key (d, n), modulo n. We can represent the decryption procedure as

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

In the RSA algorithm, the encryption key is the pair of positive integers (e, n) and the decryption key is the pair of positive numbers (d, n). Each user makes the encryption key public, keeping the corresponding decryption key private. To choose the encryption and decryption keys for the RSA algorithm, we first compute n as the product of two very large, random primes p and q. We then choose d to be a large

integer that is relatively prime to $(p-1)(q-1)$. That is, choose d such that it satisfies

$$gcd(d,(p-1)(q-1)) = 1.$$

Finally, we choose the value of e such that it satisfies the equation

$$e \cdot d \equiv 1 \pmod{(p-1)(q-1)}.$$

In addition to allowing secure encryption of messages, the RSA method of encryption also allows messages to be "signed" by the person sending the message so that the recipient has proof that the message came from the sender and not simply from someone claiming to be the sender. This is done as follows: If Alice would like to send a message M to Bob, she first uses her own private decryption procedure D_A on the message to obtain a value for $D_A(M)$. Then, using Bob's publicly available encryption procedure, she encrypts the result of the previous step to obtain the ciphertext $C = E_B(D_A(M))$. Then, when Bob receives the message he can decipher it by first using his private decryption procedure and then applying Alice's publicly available encryption procedure to obtain the original message. Formally, this can be expressed as

$$\begin{aligned} \text{First, Bob applies his own decryption procedure, } D_B(C) &= D_B(E_B(D_A(M))) \\ &= D_A(M). \\ \text{Then Bob applies Alice's encryption procedure, to get } E_A(D_A(M)) &= M \end{aligned}$$

This ability to easily represent signatures made RSA cryptography particularly well-suited for use with e-mail.

A (Simple) Example of RSA

In this section, we present a simplified example of encryption using the RSA algorithm. Consider the case where we choose the following values for p, q, n, and d: $p = 53$, $q = 61$, $n = p \cdot q = 53 \cdot 61 = 3233$ and $d = 2753$. We can compute e the "multiplicative inverse" of d to get $e = 17$.

Suppose we would like to encipher the message, "math is the coolest," using our encryption key and the RSA algorithm. Then, following the example presented in [7], we can represent the message as a number by encoding two letters per block and substituting a two-digit number between 00 and 26 for each letter (where 00 = blank, 01 = 'a', ... , 26 = 'z'). The message then becomes

$$M = 1301\ 2008\ 0009\ 1900\ 2008\ 0500\ 0315\ 1512\ 0519\ 2000.$$

If we let M_1 be the first block of M, that is $M_1 = 1301$, we can encipher the message block by block to obtain:

$$\begin{aligned} E(M_1) \equiv 1301^{17} \equiv 1301^{16} \cdot 1301 &\equiv 2230 \pmod{3233} \\ E(M_2) \equiv 2008^{17} &\equiv 2038 \pmod{3233} \\ &\vdots \\ E(M_{10}) \equiv 2000^{17} &\equiv 2698 \pmod{3233} \end{aligned}$$

Thus, we find that the enciphered message can be represented as

$$E(M) = 2230\ 2038 \ldots\ 2698.$$

To decode the message, we could similarly raise each block of $E(M)$ to the power of $d = 2753$ to obtain the orginal message.

Security of the RSA

The security of the RSA algorithm and messages encrypted using the algorithm relies on the difficulty of factoring the value of n. If n could be easily factored into the corresponding values of p and q, then one could easily find the value of d. If Marvin wanted to intercept a message that Bob sent to Alice that had been encrypted using Alice's public encryption procedure E_A, even though Marvin might be able to intercept the ciphertext $C = E_A(M)$, without the decryption key d, Marvin is unable to retrieve the orginal message. The security of the RSA algorithm can be described by **the RSA problem** and **the RSA assumption**.

The RSA Problem

The RSA problem is, given an RSA public key (e, n) and a ciphertext $C = M^e \pmod{n}$, to compute the original message, M [8].

The RSA Assumption

The **RSA Assumption** is that the RSA Problem is hard to solve when n is sufficiently large and randomly generated and the value of M (and by extension the value of C) is a random integer between 0 and $n - 1$. The RSA assumption can be thought of as the assumption that the RSA function given the choice of n and M is, in fact, a trap-door one-way function [8].

One variant on the RSA assumption that was first by Baric and Pfitzmann in 1997 is the **strong RSA assumption**. The assumption here is similar to the RSA assumption except that Marvin can select the public exponent e. That is, Marvin's task becomes: given a value for n and a ciphertext C, to compute any value of M and e such that $C \equiv M^e \pmod{n}$. This task may be easier than the original RSA problem, because Marvin is allowed to determine the value of e as well. Thus, the assumption that the task is hard to solve is a stronger one than the original RSA assumption. When we say "hard to solve," we mean that there is no efficient, polynomial-time algorithm for solving the problem.

The Math Behind the RSA Algorithm

The mathematics behind the RSA algorithm are simple, yet elegant. The algorithm works by exploiting concepts from number theory, including the properties of modular arithmetic and Fermat's Little Theorem.

The proof of the correctness of the RSA algorithm uses number theory to conclude that indeed,

$$M \equiv D(E(M)) \pmod{n} \text{ and } M \equiv E(D(M)) \pmod{n},$$

where M is the message being encrypted, E is the public encryption procedure (which encludes a public key (e, n) and an encryption method—in this case $E(M) \equiv M^e \pmod{n}$, and D is the decryption procedure (which includes the private decryption key (d, n) and the decryption procedure, here: $D(M) \equiv C^d (mod$ n) where C is the cyphertext encryption of some message. Since n is computed as a product of two large primes, p and q, and d is determined to be a large integer relatively prime to $(p-1)*(q-1)$, it is extremely difficult, given the difficultly of factoring large numbers, to compute d from e.

We define $\phi(n)$ to be the **Euler phi function** or the **totient function**, which is defined as the number of positive integers not exceeding n, which are relatively prime to n [7]. For any prime number p, $\phi(p) = p - 1$. Further, if m and n are relatively prime, then $\phi(m)\phi(n) = \phi(mn)$.

To prove that the RSA algorithm is correct, we begin by proving Fermat's Little Theorem and then using the theorem to establish the desired result.

Fermat's Little Theorem

Statement: Let p be a prime number and a an integer. Then

$$a^p \equiv a \pmod{p}.$$

Furthermore, if a is not divisible by p (that is, $gcd(a, p) = 1$), then

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: We can list the first $p-1$ positive multiples of the integer a as

$$a, 2a, 3a, \ldots, (p-1)a.$$

Suppose that $ra \equiv sa \pmod{p}$. This implies that $r \equiv s \pmod{p}$ However, since we chose distinct values for the coefficients above and r and s are both less than p, it cannot be the case that $r \equiv s \pmod{p}$. Thus we know that all of the $p-1$ multiples of a listed above are distinct and nonzero. Further, they must be equivalent (mod p) to $1, 2, 3, ..., (p-1)$ in some order. Then if we multiply the congruences, we get

$$\begin{aligned} a \cdot 2a \cdot 3a \cdot \ldots (p-1)a &\equiv 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (p-1) \pmod{p} \\ a^{p-1}(p-1)! &\equiv (p-1)! \pmod{p} \end{aligned}$$

Finally, if we divide both sides by $(p-1)!$, we arrive at the desired result [4].

■

Proof of the Correctness of RSA

The RSA Algorithm: If we represent a message as an integer M between 1 and n where n is the product of two prime numbers p and q, and

$$E(M) \equiv M^e \pmod{n} \text{ and } D(M) \equiv M^d \pmod{n}$$

where d is chosen such that $gcd(d, (p-1)(q-1)) = 1$ and $e \cdot d \equiv 1 \pmod{(p-1)(q-1)}$, then:

$$D(E(M)) \equiv M \pmod{n} \tag{1}$$

$$E(D(M)) \equiv M \pmod{n} \tag{2}$$

Proof: The left-hand sides of equations (1) and (2) can both be expressed as $(M^e)^d = M^{ed} = (M^d)^e$. Thus, to prove the correctness of the algorithm, it suffices to show [7] that

$$M^{ed} \equiv M \pmod{n}$$

Letting $\phi(n)$ represent the totient of n, we know that $e \cdot d \equiv 1 \pmod{\phi(n)}$. This implies that for some value of k,

$$M^{ed} \equiv M^{k\phi(n)+1} \pmod{n}.$$

Further, by Fermat's Little Theorem, we know that if M is not divisible by p,

$$M^{p-1} \equiv 1 \pmod{p}.$$

Therefore, since $(p-1)$ does divide $\phi(n)$,

$$M^{k*\phi(n)} \equiv 1 \pmod{p}$$

which implies that

$$M^{k\phi(n)+1} \equiv M \pmod{p}$$

An analogous argument for q gives us that

$$M^{k\phi(n)+1} \equiv M \pmod{q}.$$

Since both p and q divide n, these last two equations together imply

$$M^{k\phi(n)+1} \equiv M \pmod{n}$$

which, in turn, gives us the desired result that

$$M^{ed} \equiv M \pmod{n}.$$

■

The Significance of the Algorithm

Prior to the advent of the Internet, encryption was in many ways considered an issue only for government agencies. The RSA algorithm was introduced at a time when the potential popularity of the Internet was first becoming clear. With this popularity came a high demand for safely and securely being able to transmit information. The RSA algorithm, perceived as a nearly unbreakable, public-key cryptosystem, quickly became the method of choice for Internet cryptography including e-mail encryption among other uses. Today, RSA continues to be employed for enciphering e-mail messages as well as for the Secure Socket Layer (SSL) protocol used in the majority of internet data exchanges [6]. Thus, the RSA algorithm is something that most of us rely on each and every day, though very few of us give a second thought to the security of the e-mails we send.

Attacks on the RSA Algorithm

As we have seen, the RSA algorithm is a very secure cryptosystem that has been used for the past thirty years to provide security in millions of applications on the Internet. However, the algorithm has suffered a number of crypographic **attacks** or attempts to find and exploit weaknesses in the algorithm. Recall, the basis of the security of the RSA algorithm is that, given n it is impossible to factor n into the corresponding values of p and q in polynomial time. However, this basic assumption has neither been proved nor disproved. If there were to be a polynomial-time algorithm for factoring large numbers, the security of the internet as we know it would be compromised.

Types of Attacks

In his, "Twenty years of attacks on the RSA cryptosystem," Dan Boneh categorizes attacks on the RSA cryptosystem into four distinct classes: (1) elementary attacks that exploit blatant misuse of the algorithm, (2) low private exponent attacks, (3) low public exponent attacks, and (4) implementation attacks [1].

Elementary Attacks

Elementary attacks on the RSA algorithm rely on exploiting blatant misuse of the system. One example of this is choosing the same value of n for all users in the public file. Perhaps this seems like a good option because it is easier than computing a different value of n for each user in the system. Instead, each user can be provided unique values for e_i and d_i. However, once Marvin has a value for e_m and d_m, he can easily derive the value of n, and using Alice's publicly available exponent e_a, he is able to easily compute a value for d_a. This in fact gives us that no two users in the system should share the same value for n.

Low Private Exponent

The benefit of choosing a low private exponent is that it reduces the time required for decrypting a message. Indeed, decrypting a message is linear in $\log_2 d$. However, using a value of d that is fewer than 256 bits long leads to a system in which d can be efficiently recovered from n and e [1].

Low Public Exponent

When implementing the RSA algorithm, one may be tempted to use a low public exponent e to reduce

the time required for encryption or signiture-verification. However, attacks by Coppersmith and Hastad underscore the importance of choosing a large value for e [1]. In the following section, we will present a proof that if a low public exponent is used, then if Bob sends Alice two related messages that have been encrypted using the same values of n and e, and Marvin is able to intercept the ciphertexts of the two messages, then he is able to recover the values of the two messages.

Implementation Attacks

The final class of attacks on the RSA algorithm does not have to do with attacking the algorithm itself, but rather involves finding weaknesses in the implementation of the algorithm. One type of implementation attack is called a "timing attack" because it relies on determining the time it takes for the algorithm to perform a decryption and using this along with information about the computer with which the algorithm was implemented to determine the value of d.

One final method of attacking the RSA cryptosystem worth mentioning (though it does not cleanly fit into any of our defined classes) is called the "Man in the Middle Attack." In this situation, communication between Alice and Bob is intercepted by Marvin before it begins and Marvin is able to give both Alice and Bob his own encryption key (e_m, n_m) so that when Alice encrypts a message that she is trying to send to Bob, she actually uses Marvin's encryption key, thinking that it is Bob's. Marvin then receives the enciphered message, decipher's it using his private decryption key, and re-encrypts the message with Bob's encryption key before passing the message along to Bob. In this case, Marvin is able to read messages sent between Alice and Bob that both of them believe are private [6].

The Math Behind Some Specific Attacks

In this section, we explore two specific attacks on the RSA cryptosystem and the math behind them. Both attacks are examples of low public exponent vulnerabilities of the algorithm. The first attack, called a partial key exposure attack, shows that if a low public exponent is used and an adversary is able to partially expose some bits of the private key d, then they will be able to efficiently factor n.

Partial Key Exposure Attack

Statement: Let (d, n) be a private decryption key for the RSA algorithm and let n have N bits.

Then, given the $\lceil \frac{N}{4} \rceil$ least significant bits of d, Marvin can recover all of d in time linear in $e \log_2 e$.

Proof: The proof of the partial key exposure attack follows from a proof of the following theorem proven by Coppersmith [2]: Let $n = pq$ be an N-bit RSA modulus. Then, given the $\frac{N}{4}$ least significant bits of p or the $\frac{N}{4}$ most significant bits of p, one can efficiently factor n. From the definition of e and d, we know that

$$e \cdot d \equiv 1 \pmod{\phi(n)}$$

and therefore, that there exists k such that

$$\begin{aligned} ed - k\phi(n) &= 1 \\ ed - k((p-1)(q-1)) &= 1 \\ ed - k(pq - p - q + 1) &= 1 \\ ed - k(n - p - q + 1) &= 1 \end{aligned}$$

We know that $d < \phi(n)$, which implies that the value of k must be such that $0 < k \leq e$. Then if we reduce the above equation modulo $2^{N/4}$ and substitute N/p in for q, we obtain the following expression

$$ed - k(n - p - \frac{n}{p} + 1) = 1 \pmod{2^{N/4}}.$$

If we then multiply both sides of the equation by p, we find

$$\begin{aligned} (ed)p - kp(n - p - \frac{n}{p} + 1) &= p \pmod{2^{N/4}} \\ (ed)p - kp(n - p + 1) + kn &= p \pmod{2^{N/4}} \end{aligned}$$

Then, if Marvin is given the $N/4$ least significant bits of d, he can determine the value of $ed \pmod{2^{N/4}}$. Thus, he obtains an equation for d in terms of the values of k and p. Then, for each of the e possible values of k, Marvin is able to solve the equation for p, obtaining a number of candidate values for $p \pmod{2^{N/4}}$. Then, for each of these values, Marvin can run the efficient algorithm guaranteed by Coppersmith's theorem above to attempt to factor n. The number of candidate values for p is at most $e \log_2 e$. Thus we have shown that even if only a portion of d is revealed, the security of the system is compromised if the value of e is chosen to be low [1].

■

Related Message Attack

In a related message attack, if Bob sends Alice related messages that have been enciphered using the same value of n for both, if Marvin intercepts the ciphertext of the two messages, he may be able to recover the original messages if the value of the public exponent e is low. The attack works for any low value of e, but, to simplify the proof, we choose $e = 3$. The proof of the related message attack relies on using the a modified version of the **Euclidean algorithm**, an iterative algorithm for determining the greatest common divisor of two values, which is linear in the size of the smaller of the two integers. In this case, we must use the Euclidean algorithm for polynomials.

Statement: Let $e = 3$ and let (e, n) be an RSA public encryption key. Let two messages M_1 and M_2 such that $1 < M_1, M_2 < n - 1$ and $M_1 \neq M_2$ also satisfy $M_1 = f(M_2)$ mod n for some linear polynomial $f(x) = ax + b \in \mathbb{Z}_n[x]$ with $b \neq 0$. Then, given the values of n, e, C_1, C_2 and the expression for f, one can recover M_1 and M_2 in time quadratic in $\log n$.

Proof: By the RSA algorithm, we know that $C_1 \equiv M_1^e \pmod{n}$. Thus, since M_2 is a root of the equation $f(x) - M_1 \pmod{n}$, M_2 is a root of the polynomial

$$g_1(x) = f(x)^e - C_1 \in \mathbb{Z}_n[x].$$

Similarly, since $C_2 \equiv M_2^e \pmod{n}$, we know that M_2 is a root of the polynomial

$$g_2(x) = x^e - C_2 \in \mathbb{Z}_n[x].$$

Both polynomials are divisible by the linear factor $(x - M_2)$ and therefore one can use the Euclidean algorithm[1] to compute the greatest common divisor of g_1 and g_2. Then, if the gcd of the two polynomials is linear, then M_2 has been found (and, since $M_1 = f(M_2) \pmod{n}$, M_1 has also been found).

When $e = 3$, the greatest common divisor of e and $\phi(n)$ is 1 and thus there is only one root of the polynomial $g_2 = x^3 - C_2$ that is in $\mathbb{Z}_n$. Therefore, g_2 factors modulo both p and q into a linear factor and an irreducible quadratic factor. Then since g_2 does not divide g_1, their greatest common divisor

[1] While, technically, $\mathbb{Z}_n[x]$ is not a Euclidean ring, if the algorithm fails here, then we get a nontrivial factor of n. Each time we have to divide modulo n, we use the Euclidean algorithm to find the inverse modulo n. Then, if at any stage the algorithm fails to provide an inverse, we have either obtained a nontrivial divisor of n or we obtain n itself as the greatest common divisor. In the former case, we have exposed the factorization of n. In the latter, we can continue using the Euclidean algorithm to find the factorization of n [5].

must be linear, and so if $e = 3$, one can determine the values of M_1 and M_2 if given the values of the corresponding ciphertexts [1].

■

Future of the RSA Algorithm and Public-Key Cryptography

Attacks on the RSA will continue to get stronger as factoring algorithms are improved and made faster. There are many ideas for improving RSA security. First and foremost is the hope that simply choosing increasingly longer keys will make the factorization problem more difficult and help prevent attacks. As few as five years ago, RSA encryption that used a 512-bit value for n was considered safe. Now, improvements in technology have made it so that a 512-bit RSA system can be broken in just a few days [6]. As computers continue to increase their computing power, increasing key length will continue to be a good solution against many of the attacks presented here. The main problems that RSA and other encryption mechanisms must be prepared to deal with include improved computing speed and capacity and mathematical breakthroughs for factoring large numbers.

Conclusions

In this paper, we have described the famous RSA algorithm for public-key cryptography. The algorithm, which was developed in 1977 by Rivest, Shamir, and Adlemen, has become one of the most widely-used cryptography systems since it was adopted for enciphering e-mail messages and other tasks involving security on the Internet. We have seen that RSA is, at its core, a piece of simple mathematics which makes use of facts and theorems from number theory, including Fermat's Little Theorem. We have looked at several kinds of attacks both on the algorithm itself and on implementations of the algorithm. Finally, we recognize that as computing capacity increases, the future of public-key cryptography will necessarily involve using longer decryption keys to maintain security.

References

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

[2] D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *Journal of Cryptology*, 10:233–260, 1997.

[3] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[4] G.H. Hardy and E.M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, 1960.

[5] Neal Koblitz. *A Course in Number Theory and Cryptography*. Springer, 1994.

[6] B. Morgan and D. Grimshaw. The dangers of putting too much trust in RSA, 2003.

[7] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital-signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[8] H.C.A. van Tilborg. *Encyclopedia of cryptography and security*. Springer, 2005.