PROVABLY SECURITY FOR *OAEP*+-RSA AND *SAEP*-RABIN

Laurent Floor, 5533937
Supervised by prof. dr. G. Cornelissen

15 june 2018

Universiteit Utrecht

A thesis submitted for the bachelor Mathematics, TWIN

Contents

1 Introduction

In modern communication cryptography plays an essential role. Digital messages contain not only private but also valuable information. Using public-key cryptography we are able to protect our bank transactions and mail traffic against eavesdroppers. This stresses the importance of having the most secure protocol to encrypt our digital messages.
To find the most secure encryption protocol we must compare security of different protocols.
This thesis is about a mathematical description of provable security of communication. The thesis is divided into three parts. First we give the definitions of public-key cryptography and provable security. In the second part two important encryption functions and their mathematical background are given. Finally, two encryption protocols, based on the encryption functions, are discussed and their security is proven. In this way we are able to give a statement which protocol is preferable to use.

Public key cryptography and provable security

In chapter 2 we give a short introduction to the conventions in cryptography. We restrict ourselves this thesis to an active adversary in a public-key cryptography setting. This means that the receiver generated a public key for encrypting and private key for decrypting. The adversary, the one that tries to break the encryption protocol, is allowed to ask decryptions from the receiver that knows the private key.
After the outline about cryptography we give in chapter 3 a definition for the security of a protocol. A message is encrypted secure if from the ciphertext we can get no information about the message. This is formalized in definition 3.2 of indistinguishable security.

RSA and Rabin

Before we apply this definition to some protocols we introduce two important encryption-functions. In chapter 5 we discuss the well-known RSA encryption-function. Secondly, we treat in chapter 6 encryption with the Rabin-function. The mathematical background behind this cryptographic primitives is treated in chapter 4.
It appears that both encryption-functions are vulnerable to attacks if used naively. Therefore, the functions will be combined with a padding-protocol which randomizes the function before encryption.

Cryptographic protocols and their security

In chapter 7 and 8 we give the cryptographic protocols $OAEP+$-RSA and $SAEP$-Rabin, the combination of the padding-protocols $OAEP+$ and $SAEP$ with the RSA respectively Rabin function. For both protocols we give a security proof in the form of a reduction argument. This means that breaking the security of the protocol implies that a non-efficient solvable mathematical problem can solved fast. Factoring a integer N into primes reduces to $SAEP$-Rabin and finding the inverse of $f(x) = x^e \mod N$ reduces to $OAEP+$-RSA.
Comparing the security claims of the protocols in chapter 9 we find that the $SAEP$-Rabin is preferred to use. In practice, most times the RSA-function is used. According to Koblitz and Menezes [9] this stress the importance of accessibility and understandability of the field of provable security.

I thank my supervisor prof. dr. G. Cornelissen for providing me with the interesting subject of this thesis and giving me valuable suggestions to explore the subject.

2 Introduction to cryptography

Cryptography is all about the question if we are able to send messages safely. In this chapter we give a brief introduction to the most important features of cryptography. First we set up the scheme of communication. Then we give a short description about the distinction between symmetric and asymmetric cryptography.
For the basis of communication we need a sender and receiver. Literature calls the sender of the messages Alice and the receiver Bob. Introducing the adversary Charly[1], we have to search for a protocol such that Charly is not able to intercept information contained in the message. To make this possible Alice and Bob have agree on an encryption protocol E for Alice and a decryption protocol D for Bob. Those protocols contain at least an encryption and decryption function: $f_{encr,K}$ and $f_{decr,K}$ respectively, which depends on key information K.

2.1 Symmetric-key cryptography

The most simple way to encrypt and decrypt messages is when the sender and receiver have the same key. In this symmetric-key system Alice and Bob use the same key to execute their protocols. For example, we consider the Caesar Cipher. Given a message m, the sender shift every character of m by k places in the alphabet to get the ciphertext. For the receiver is it easy to trace back the original message by shifting every character k places back.
The core of symmetric-key cryptography is that both Alice and Bob have to keep the same key private. Therefore, it is also called private-key cryptography. There are multiple ways to design and perform symmetric-key cryptography. But in every case we get into the same difficulties. There are two important problems using symmetric-key cryptography. First the question arises how Alice and Bob agree on the key-information. In earlier times there were possibilities to exchange the information physically or personally. Nowadays, in a time of worldwide communication it is nearly impossible to transfer keys in an secure way. Secondly, as in the case of the Caesar Cipher, it is often easy to get information about the key from the ciphertext. Counting how often some characters appear and using knowledge how often these letters appear in texts you can reduce information about the plaintext. To solve this you can use for every character a different replacement. This has the disadvantage that you need a very long key to encrypt and decrypt messages.

2.2 Asymmetric-key cryptography

Because of these drawbacks, asymmetric cryptography is used today. In protocols using this principle the receiver Bob generates two kinds of key information, first the public information key K_e for Alice to encrypt messages and secondly the private information key K_d is kept secret to decrypt ciphertext. In cryptography, we search for functions that satisfy the condition $f_{encr,K_e} = f^{-1}_{decr,K_d}$. These functions we call the primitives. Besides this, the most important requirement is that an adversary Charly cannot execute f_{decr,K_d} unless he knows K_d. The meaning of this is that in the eyes of Charly the encryption primitive is a one-way function. Before introducing some examples of such primitives, we give a definition for the security of the protocols.

[1]In contrast to Alice and Bob, the adversary has no fixed name. This varies in literature from Marvin [3] for a malicious attacker to Eve for an eavesdropper.

3 What is provable security?

In this section we give a mathematical definition of provable security. In the next chapters this definition will be applied to some cryptographic protocols, like RSA. First we stress the importance of an appropriate definition. Thereafter, we introduce the mathematical definition of provable security. We conclude this section by explaining the way in which this definition is usually used to claim the security of the protocol.

3.1 Importance

Answering the question when a cryptographic protocol is secure, is more difficult than it seems at first glance. The simple answer says that an encrypted message is secure when an adversary is not able to get any information about the message from the cipher text. Or stated conversely, a protocol is secure if all kinds of attacks are fruitless. In this sense attacks and security are opposite sides of the same coin and are each others complements. [1, Section 3.3] In practice we see in the course of time that new weaknesses are found in protocols and the security is being at stake.
This emphasizes the importance for a formal definition of provable security. Such a definition says under which conditions and against which attacks a cryptographic protocol is secure. In this way the strength and weakness of each system is clear. We then have the opportunity to discuss and compare the security of different protocols. [11]

3.2 Definition

Before we give a formal definition of provable security, we want to introduce two kinds of adversaries. Their difference is based on the abilities the attackers have. An adversary can be passively breaking the protocol, only making use of the public information of the receiver and a challenged ciphertext c. On the other hand, the attacker can actively ask information of the receiver. [11, Page 6]

Definition 3.1 (Active adversary)**.** *An attacker of a protocol is an active adversary if he is allowed to ask decryption of chosen ciphertexts not equal to c.*

An active adversary can forge a Chosen Ciphertext Attack (CCA). It follows that if a system is secure against CCA, also a passive adversary is not able to break the security of the protocol with success. We restrict ourselves only to active attacks.
An informal definition of provable security for public-key cryptography goes as follows. Let Alice encrypt her message m by the public key information K_e of Bob to get the ciphertext $y = f_{encr,K_e}(m)$. We can say that this message is secure against CCA if it is (for an active attacker Charlie) not feasible to get any information about m by analysing y, even if he may request for decryptions of ciphertexts not equal to y. In this notion, called semantic security, the concept of information is not described. This solution for this problem is elaborated in the definition of indistinguishable security. It is proven that both definitions are equivalent for active and passive adversaries. [9, Section 1.3]
In the definition of indistinguishable secure we look at the probability for Charley to have success for choosing the correct message from a ciphertext. Charley has two plaintext messages m_1 and m_2. Randomly, one of these message is encrypted with ciphertext y^*. Charley breaks the security if he can decide which message is encrypted with a probability greater than 50%. [9, Section 1.3] [11, Chapter 3]

Definition 3.2 (Indistinguishable security)**.** *Given $\epsilon > 0 small$, an encryption protocol E is indistinguishable secure against Chosen Chiphertext Attacks if an active adversary has an algorithm A with executing time t to determine which of the two messages is encrypted with a chance equal or less than* $1/2 + \epsilon$.

3.3 Reduction argument

In cryptographic analyses the word 'proof' is a misleading term. A security proof suggest a 100 percent guarantee that a system is safe to use. History teaches us the fallacy of this statement. So we better talk about arguments that support a security claim. In cryptographic analyses reduction

arguments are the most common. In this kind of argument we compare a cryptographic system with an unsolvable or a very hard mathematical problem. For instance, breaking Rabin is as hard as factoring a big integer N.
We say a mathematical problem P_1 reduces to a cryptographic problem P_2 if the hardness of P_1 implies the hardness of P_2. Suppose we have an adversary A who breaks the cryptographic problem (for example Rabin). If we can use A to build another algorithm B to solve the mathematical problem (as factorization), P_1 reduces to P_2. After centuries of research we expect that factorization can not be done in an efficient way, hence we claim that there is also no efficient algorithm to solve Rabin. [1, Section 2.2] [10, Section 5] In the security proof we analyze the differences between A and B. We assumed that A solved the cryptographic protocol E. Following definition 2 given in the previous paragraph, the adversary succeeded in time t_A with probability of success greater than $1/2 + \epsilon$.
For algorithm B we get an execution time t_B with probability of success ϵ_B. The assumption that B solves a mathematically hard problem gives that t_B is large. The step where A reduces to B needs $t_R = t_B - t_A$ time. This is the reduction time. To study t_B and t_R we can give a statement about t_A. For example, we assumed that t_B is large and if we find that the reduction step is very fast hence t_R is small, gives that the adversary needs much time to break the cryptographic problem. This we summarize in the following definition: [8, Definition 9.17] [9, Page 7]

Definition 3.3 (Reduction argument). *Assume algorithm A solves the cryptographic protocol E in time t with a chance of success greater than $1/2 + \epsilon$. A mathematical problem P_1 reduces to E if there exist an polynomial algorithm B that solves from A the problem P_1 in time t_B.*

Remark. The reduction time is bounded by the magnitude of input size n, $t_R = O(f(n))$ for some function f. If the reduction step is small, for example $t_R = \log n$ or $t_R = n$, we have a tight reduction. Otherwise we have a non-tight reduction. We prefer to have a tight reduction, because then we are more safe to argue that t_A is big. [10, Section 4]

4 Mathematics behind cryptography

Before we get into the examples of cryptographic primitives, some mathematical background is given. We give the most important theorems so that we can smoothly discuss the RSA and Rabin functions in chapter 5 and 6. In general we use the mathematical knowledge as given in [8] and [15, Chapters 5 and 6].

4.1 The multiplicative group

For the mathematical background of the RSA and Rabin primitive we need to have a brief introduction to modular arithmetic. First, we define the modular group $\mathbb{Z}_N$ as the equivalence classes of integers modulo N. Integers a and b are equivalent if they are congruent $a \equiv b \mod N$. This group is closed under addition.
For RSA we need a group which is closed under multiplication. This group, a subset of $\mathbb{Z}_N$, is defined in the following way.

Definition 4.1 (Multiplicative group)**.** *The multiplicative group modulo N is a group of equivalence classes of integers given by $\mathbb{Z}_N^* = \{a \in \mathbb{Z}_N | \exists a' \in \mathbb{Z}_N : a \cdot a' = 1\}$.*
The number of elements is defined as the Euler's totient $\varphi(N) = |\mathbb{Z}_N^*|$.

The set-up of this group can also be given by looking at the divisors of the modulus. The greatest common divisor of integers $a, b \in \mathbb{N}$ is defined as the largest integer d such that d divides both a and b, notation $\gcd(a, b) = d$. Two integers a, b are relative prime if $\gcd(a, b) = 1$.

Theorem 4.1. *An element $m \in \mathbb{Z}_N$ is an element of $\mathbb{Z}_N^*$ if and only if* $\gcd(m, N) = 1$.

Proof. Suppose $\gcd(m, N) = 1$, from the Euclidean algorithm we get $lm + kN = 1$ for some integers k, l. This implies that $lm + kN \mod N = lm \mod N = 1$. We see that the inverse of m is l. Hence $m \in \mathbb{Z}_N^*$.
For the other direction, we assume that $m \in \mathbb{Z}_N^*$, and thus we have $m \cdot m' = 1 \mod N$. This gives us that for some k we can write $mm' + kN = 1$, which implies that $\gcd(m, N) = 1$. □

4.2 Multiplicative group and primes

In this subsection we explore more features of the multiplicative group. We study the special case when we choose the modulus of the group equal to a prime number. We start by studying the Euler totient. For a divisor d of n we write $d|n$.

Theorem 4.2. $\sum_{d|N} \varphi(d) = N$.

Proof. Let $Z_d = \{a | 1 \leq a \leq N \wedge \gcd(a, N) = d\}$ a set. Every element $k \in \mathbb{Z}_N^*$ belongs precisely to one Z_d, hence $N = \sum_{d|N} |Z_d|$. We construct the bijective function $g : Z_d \to \mathbb{Z}_{N/d}^*$ with $a \mapsto a/d$. We conclude by writing $N = \sum_{d|N} |Z_d| = \sum_{d|N} |\mathbb{Z}_{N/d}^*| = \sum_{d|N} \varphi(d)$ □

If we choose the modulus of $N = p^k$ then we can easily calculate the result of the Euler's totient-function.

Corollary 4.1. *If $N = p^k$ then $\varphi(p^k) = (p-1)p^{k-1}$.*

Proof. From the previous theorem we can write that $\varphi(p) + \varphi(p^2) + ... + \varphi(p^k) = p^k$ and subtract this from $\varphi(p) + \varphi(p^2) + ... + \varphi(p^{k-1}) = p^{k-1}$, giving $\varphi(p^k) = p^k - p^{k-1} = (p-1)p^{k-1}$. □

Another mathematical result we need, we get from the Lagrange Theorem on groups. We restrict ourselves to Abelian groups.

Theorem 4.3. *For a finite Abelian group G with unit element e, for every $x \in G$ we have $x^{|G|} = e$.*

Proof. We construct the bijective function which maps the group to itself $G \to G, g \mapsto xg$. Now we write $\prod_{g \in G} g = \prod_{g \in G} gx = x^{|G|} \prod_{g \in G} g$, which implies the equality. □

We apply this theorem to the multiplicative groups in the following theorems.

Theorem 4.4. *For a relatively prime to N we have $a^{\varphi(N)} = 1 \mod N$.*

Proof. If a is relatively prime to N, then $a \in \mathbb{Z}_N^*$. Now we use theorem 4.3 to get $a^{\varphi(N)} = a^{|\mathbb{Z}_N^*|} = 1 \mod N$. □

For a group with modulus equal to a prime, from Theorem 4.4 Fermat's Little Theorem follows.

Corollary 4.2 (Fermat's Little Theorem)**.** *For p is prime such that $gcd(a, p) = 1$ we get $a^{p-1} = 1 \mod N$*

4.3 Chinese Arithmetic

The RSA and Rabin-primitives use the modular group with modulus $N = p \cdot q$ with p, q primes. Using the Chinese Remainder Theorem we can map elements in $\mathbb{Z}_p$ and $\mathbb{Z}_q$ to $\mathbb{Z}_N$.
First we define $W_p, W_q \in \mathbb{Z}_N$. Because $\gcd(p, q) = 1$, from theorem 4.1 it follows that there is an integer $y_p \in \mathbb{Z}_q$ such that $p \cdot y_p = 1 \mod q$. Let $W_q = p \cdot y_p \mod n \in \mathbb{Z}_N$. Similarly, we construct $W_p = q \cdot y_q \mod N \in \mathbb{Z}_q$ with y_q such that $q \cdot y_q = 1 \mod p$.

Theorem 4.5 (Chinese Remainder Theorem)**.** *If p and q are primes and $p \neq q$, the function $f : \mathbb{Z}_p \times \mathbb{Z}_q \to \mathbb{Z}_N, (a_1, a_2) \mapsto a = a_1 \cdot W_p + a_2 \cdot W_q \mod N$ maps every element in $\mathbb{Z}_p \times \mathbb{Z}_q$ unique to an element in $\mathbb{Z}_N$.*

Proof. Let $a = a_1 \cdot W_p + a_2 \cdot W_q \mod N$, then $a \mod p = a_1 \cdot 1 + a_2 \cdot 0 = a_1$ and $a \mod q = a_1 \cdot 0 + a_2 \cdot 1 = a_2$. Hence we can define the inverse of f as $f^{-1}(a) = (a \mod p, a \mod q)$.
Further notice that f is injective and surjective, hence f is a homomorphism. The number of elements in the groups are equal: $|\mathbb{Z}_N| = N = p \cdot q = |\mathbb{Z}_p| \cdot |\mathbb{Z}_q| = |\mathbb{Z}_p \times \mathbb{Z}_q|$, hence every element in $\mathbb{Z}_p \times \mathbb{Z}_q$ maps to an unique element in $\mathbb{Z}_N$. □

5 The RSA function

One of first and most well known primitives is the so called RSA-function. The name is derived from the scientists who proposed the primitive in 1977: Rivest, Shamir and Adleman.

5.1 The RSA primitive

We describe the operation of the RSA-primitive. [13] Choose the modulus $N = p \cdot q$, where p and q are large prime numbers. Now we have the multiplicative group $\mathbb{Z}_N^*$ with order $\varphi(N) = (p-1)(q-1)$. Let e and d be two integers satisfying $e \cdot d = 1 \mod \varphi(N)$. The modulus N and the encryption exponent e are the public information of the RSA system. We define the RSA primitive as

$$f_{encr,e}(x) = x^e \mod N.$$

To decrypt the ciphertext we use the inverse

$$f_{decr,d}(x) = x^d \mod N.$$

Given a message $m \in \mathbb{Z}_N$ we can verify the operation of the primitive. After encryption we get the ciphertext $c = m^e \mod N$. For deciphering it is intuitively seen that $c^d = m^{de} = m$, but we make it explicit in the following claim.

Claim 5.1. *Applying $f_{decr,d}$ to ciphertext $c = m^e$ gives the plaintext m.*

Proof. Given is the encrypted message $c = m^e$, the public and private key. We have chosen these keys such that $e \cdot d = 1 \mod \varphi(N)$. This implies there is an integer k such that $e \cdot d - 1 = k(p-1)(q-1)$. If $\gcd(p, m) = 1$ we can write from Fermat's Little theorem $m^{ed} = m^{ed-1}m = m^{k(p-1)(q-1)}m = (m^{p-1})^{k(q-1)}m = m \mod p$. In the same we say if $\gcd(q, m) = 1$ then $m^{ed} = m \mod q$.
Now we assume $\gcd(p, m) \neq 1$. Because p is prime $m = 0 \mod p$ and therefore $m^{ed} - m = 0 \mod p$. In the same way if $\gcd(q, m) \neq 1$ then $m^{ed} = m \mod q$.
It follows that m^{ed} is multiple of q and p. From the Chinese Remainder Theorem 4.5 it follows that for some l we have $m^{ed} - m = l \cdot p \cdot q$ gives $m^{ed} = m \mod N$ □

6 The Rabin primitive

Another asymmetric cryptographic system is the Rabin cryptosystem. Two years later than RSA, Rabin published his technique. [12] The security of the Rabin encryption protocol is based on the assumption that factorization is difficult. In this section we discuss the operation of the Rabin primitive.

6.1 The Rabin primitive

The Rabin function is a special case of the RSA-function. Where in RSA the key generated also the encryption exponent, in Rabin the exponent is chosen to be two. Instead of finding the inverse function of the exponent, we have to find roots of the ciphertext.
Like RSA we again choose two large prime numbers to construct the modulus $N = p \cdot q$. [8] We restrict the primes in the following way: $p \equiv q \equiv 3 \mod 4$. This restriction is not necessary for Rabin, but we use it to simplify the decryption and because it is applied in practice. Given a message m, the encryption function squares m modulo N:

$$f_{encr,N}(x) = x^2 \mod N.$$

The security of Rabin is based on the assumption that for N you can not find the roots of the ciphertext, but for p and q it is possible. Hence, if you can factor an integer $N \in \mathbb{N}$ you can break Rabin using the decryption steps.
To decrypt a ciphertext $c = y^2 \mod N$ we use the Chinese Remainder Theorem 4.5 to map the solution in $\mathbb{Z}_p$ and $\mathbb{Z}_q$ to $\mathbb{Z}_N$. From the restriction on p, we see that $\frac{p+1}{4}$ is an integer modulo N. Hence we can write for c: $z_p = c^{\frac{p+1}{4}} \mod p$. When we square this, we get $z_p^2 = c \cdot c^{\frac{(p-1)}{2}} \mod p$. From theorem 4.4 we know that the last term is $c^{\frac{(p-1)}{2}} = y^(p-1) = 1 \mod p$.
If $\gcd(c,p) \neq 0$ we see that $c \in \mathbb{Z}_N^*$ and thus it has an inverse. Hence, both z_p and $-z_p \equiv p - z_p \mod p$ are the roots of c modulo p.
If $\gcd(c,p) = 0$ we see that only $z_p = 0$ is an root. In the same way we find a solution for $c \mod q$. Remark that for at most one of the prime numbers we can have a zero root.
We can then easily calculate four (or two) resulting roots using the Chinese Remainder Theorem.
To decide which of these roots is the correct message, you need more information about the message. For example the length or the content of the message. In chapter 8 we will come back on this issue.

7 Optimal asymmetric encryption padding+-RSA

In chapter 5 we described the RSA-function. In practice, it is not applicable to use an encryption system where the protocol consist of applying the primitive. Suppose the adversary has as target message m with ciphertext y encrypted with RSA and encryption exponent e. If Charly is an active adversary as in definition 3.1, he is able to ask the decryption of $y' = y \cdot m'^e$ and gets $m'm$. Then Charly can divide by m' to find the target message. [9, Page 6]
To protect us against such Chosen Ciphertext Attacks (CCA) we pad each message before encrypting. In this chapter we will describe one of the padding protocols together with the RSA primitive. Finally, we give and prove a security claim for $OAEP$+-RSA.

7.1 Padding tools

To protect our communication against CCA we have to randomize the message. We enlarge the encryption-protocol E with a modification phase P. Before the primitive is applied, the message is randomized by some tools we will describe here. Suppose we have a message m with bitlength l. This means the message is an element of set $\{0,1\}^l$. In P we construct from m a padded message y with bitlength $k \geq l$. In literature, k is called the security parameter. The ciphertext is then calculated from y by applying the primitive.
First we give some possible tools to blind m. After this we describe in the next paragraph an example of an encryption-protocol with a padding phase. The most used way to randomize m is introducing a random integer r. The sender chooses for every message a unique r. He appends this integer to y such that an adversary can not track r down if he knows only the ciphertext c.
Another important tool for P is the use of hash-functions. These are random functions where the inverse function cannot be computed. An ideal hash-function has the property that from the output we can not deduce any information about the input. In that sense, hashing is the same as encryption, with the difference that for encryption you have a decryption-function. For example in $OAEP$+ the hash-function $SHA1$ (Secure Hash Algorithm) is used. [15, Section 7.3]

Definition 7.1 (Hash-function). *A hash-function $h : \{0,1\}^{l_0} \to \{0,1\}^{l_1}$ is called ideal if given $b \in \{0,1\}^{l_1}$ it is not feasible to find a such that $f(a) = b$ without trying all possibilities.*

The last tool we discuss is adding zero-bits to m. In chapter 8 we merge m with l_0 zero-bits. We denote this concatenation with $m^0 = m||0^{l_0}$. The advantage of this tool is that after decryption the receiver can check if the input was generated in a proper way.

7.2 $OAEP$+-RSA

In this paragraph we describe the encryption-system $OAEP$+-RSA. This system consist of the padding protocol $OAEP$+ with the RSA-primitive. We notice that the padding protocol can be combined with another encryption primitive. The protocol $OAEP$+ was introduced by Shoup as a revised version $OAEP$. [14, Section 6]
First we choose the security parameter $k = l + l_0 + 1_l$ where l is the bitlength of message m and $k \leq N$, where N is the modulus of RSA. Second the protocol needs three independent ideal hash-functions:

$$\begin{aligned} &G : \{0,1\}^{l_0} \to \{0,1\}^l \\ &H_1 : \{0,1\}^{l+l_1} \to \{0,1\}^{l_0} \\ &H_2 : \{0,1\}^{l+l_0} \to \{0,1\}^{l_1} \end{aligned}$$

We define the $\oplus$ or XOR-function in the following way:

Definition 7.2 (XOR-function). *The function $\oplus : \{0,1\}^l \times \{0,1\}^l \to \{0,1\}^l$ is defined for every bit a_i and b_i of the bit strings a and b as*

$$a_i \oplus b_i = \begin{cases} 0 & \text{if } a_i = b_i = 0 \text{ or } a_i = b_i = 1 \\ 1 & \text{if } (a_i = 1 \text{ and } b_i = 0) \text{ or } (a_i = 0 \text{ and } b_i = 1). \end{cases}$$

Further the sender chooses a random integer $r \in \{0,1\}^{l_0}$. Now we can encrypt m with the following steps:

$$\begin{aligned} s &= (G(r) \oplus m)||H_2(r||m), \\ t &= H_1(s) \oplus r, \\ c &= f_{RSA,e}(y) = f_{RSA,e}(s||t). \end{aligned}$$

The decryption for ciphertext c is done by working out the next tree steps:

$$\begin{aligned} y &= s||t = f_{RSA,d}(c), \\ r &= t \oplus H_1(s), \\ m &= G(r) \oplus s_1 \end{aligned}$$

where s_1 are the first l bits of s. An ciphertext is valid if $H_2(r||m) = s_2$, the last l_1 bits of s.

7.3 Security Claim

In this paragraph we give the argument of the security of $OAEP$+-RSA . We follow the proof as given in [14, Section 6].

Theorem 7.1. *Assume G, H_1 and H_2 are ideal hash-functions. Assuming there is an adversary A who breaks OAEP+-RSA with running time t and probability of success greater than $1/2 + \epsilon$, and q_d, q_{H_1}, q_{H_2} and q_G are the number of queries made by the adversary,*
then there is an algorithm B that inverts the RSA-primitive with execution time

$$t + O(q_G q_{H_1} k^2 + (q_G + q_{H_2} + q_{H_1} + q_d)k)$$

and chance of success greater than

$$\epsilon - \frac{q_{H_2} + q_d}{2^{l_1}} - \frac{(q_d + 1)q_G}{2^{l_0}}.$$

Proof. For the proof we assume there is an adversary A who breaks $OAEP$+-RSA. Given two messages m_0 and m_1 and the decryption y^* of $m^* = m_b$ with $b \in \{0,1\}$ random chosen. Then A give as output b' with probability that $b = b'$ equal to $1/2 + \epsilon$.
We want to compare A to algorithm B. To do this we build algorithms A_1, A_2 and A_3. Between each of the algorithms we describe the reduction. Further, we deal with an active adversary. Hence the adversary queries for values of the hash-functions, asks decryptions for some ciphertexts and save this data in the sets S_{H_1}, S_{H_2}, S_G and S_q with cardinality q_{H_1}, q_{H_2}, q_G and q_d.

Difference on saved data

We construct algorithm A_1 that can decrypt ciphertext only if the parts of the decryption are in the sets S_{H_1} and S_{H_2}. The algorithm acts like the adversary, except for the difference we describe here.

1. Given a ciphertext y, the algorithm A can calculate all decryption steps properly. This is not for true A_1, it rejects y when $(r,x) \notin S_{H_2}$. To analyze the difference in the probability for this case we write the following.
   Assume A_1 has to decrypt ciphertext $y \neq y^*$. If $r = r^*$ and $m = m^*$ then $s_2 \neq s_2^*$. This will be rejected in A and so in A_1. Hence $r \neq r^*$ or $m \neq m^*$. If $H_2(r||m)$ is never queried, it has a random output c'. The probability that this is equal to c is given by $P(c = c') = \frac{1}{2^{l_1}}$. Given an total of q_d decryptions, the probability that a ciphertext is not rejected by A_1 when $(r,x) \notin S_{H'}$ is $\frac{q_d}{2^{l_1}}$.

2. Secondly we also rejects ciphertext where $s \notin S_H$. Given an ciphertext $y \neq y^*$ this splits in two cases. First we describe when $s = s^*$ and in the next case we describe when $s \neq s^*$.
   Assume A_1 has to decrypt $y \neq y^*$ with $s = s^*$. This implies $t \neq t^*$ and $r \neq r^*$. The ciphertext is valid when accidentally $H_2(r||m) = H_2(r^*||m^*)$, the probability for this is $\frac{q_{H_2}}{2^{l_1}}$.

3. Assume A_1 has to decrypt $y \neq y^*$ with $s \neq s^*$. This implies $t = t^*$. Because the adversary never queries $H(s)$, from $r = t \oplus H(s)$ we see that r is independent. The probability $r \in S_G$ is $\frac{q_d q_G}{2^{l_1}}$.

We conclude that the difference of success between both A and A_1 is the sum of the chances described in the following three cases:

$$|P(\text{Succes of } A) - P(\text{Succes of } A_1)| \leq \frac{q_{H_2} + q_d}{2^{l_1}} + \frac{q_d q_G}{2^{l_0}}.$$

Pass the primitive

Now we construct algorithm A_2 that, given ciphertext c, tries for all elements of S_{H_1} and S_{H_2} to find a root y by computing for each pair $(r', m') \in S_{H_2}$:

$$\begin{aligned} s' &= (G(r') \oplus m)||H_2(r'||m') \\ t' &= H(s') \oplus r \\ y' &= f(s'||t') \end{aligned}$$

If we find a $y' = y$ the algorithm is successful. From the definition from A_1 we see that the probability for success for A_2 is the same. Indeed, the probability that you construct from the saved sets a y equal to a random y' is the same as that you choose a random y' such that is equal to one of the y that can be made from the saved sets.
This step needs a lot of work. We build an table of (r, m), in total of $q_G \cdot q_{H_1}$ values. For every value we need T_{RSA} time to evaluate the RSA encryption function. For the RSA-primitive we need to raise y with length k to the power e. In practice an exponent e is chosen such that the complexity of the function is $O(k^2)$. [15, Section 6.1].
The time to build the table is $(q_G + q_{H_1} + q_{H_2} + q_d)k$. Gathering this, we get the final bound of the algorithm

$$O(q_G q_{H_1} k^2 + (q_G + q_{H_1} + q_{H_2} + q_d)k)$$

Conclusion

The last step is to go from A_2 to an algorithm A_3 which is fully random. We define the success probability of A_3 as $1/2$.
In the current context the difference between A_2 and A_3 is the knowledge if r^* is queried or not in A_3 though the adversary. There are two cases. The probability that r^* is queried and also in the list S_G is given by $\frac{q_G}{2^{l_0}}$. The other possibility is that r^* is not queried and hence not in S_G. That is only possible if there is an algorithm B with running time t' that with success ϵ' can find the inverse.
Now we write the final conclusion. The relation of probability of the success of A_2, A_3 and B can written as:

$$\begin{aligned} &|P(\text{Success of } A_2) - P(\text{Success of } A_3)| \leq \frac{q_G}{2^{l_1}} + P(\text{Success of inverting}) \\ &P(\text{Success of } A_1) - \frac{1}{2} \leq \frac{q_G}{2^{l_0}} + \epsilon' \\ &\frac{1}{2} + \epsilon - \frac{q_{H_2} + q_d}{2^{l_1}} - \frac{q_D q_G}{2^{l_0}} - \frac{1}{2} \leq \frac{q_G}{2^{l_0}} + \epsilon' \\ &\epsilon' \geq \epsilon - \frac{q_{H'} + q_d}{2^{l_1}} - \frac{q_D q_G}{2^{l_0}} - \frac{q_G}{2^{l_0}} \end{aligned}$$

□

8 Simple asymmetric encryption padding-Rabin

Another padding encryption protocol is the Simple Asymmetric Encryption padding, proposed by Boneh [4], combined with the Rabin primitive. In this chapter we first give the working of protocol. We apply in this case the Rabin primitive as encryption function. After this, we give a proof of the security of $SAEP$-Rabin. We follow in this chapter the discussion of $SAEP$-Rabin in [8, Section 9.5.1].

8.1 $SAEP$-Rabin

Like Rabin, described in chapter 6, we choose the modulus $N = p \cdot q$ with $p \equiv q \equiv 3 \mod 4$. To let the protocol work, we give some restriction on the length of the modulus. Let $k \in \mathbb{N}$ the security parameter. We choose p and q such that the bitlength is $\frac{k}{2}+1$. Then the modulus has length $k+2$. This means that $2^{k+1} \leq N < 2^{k+2}$.
To encrypt a message m with bitlength l, we take the following steps. First we append to the message l_0 zero bits to get

$$m^0 = m||0^{l_0}.$$

We choose a random integer r of bitlength l_1 and apply the ideal hash-function $h : \{0,1\}^{l_1} \to \{0,1\}^{l_0+l}$ in the following way to get the padding message

$$y = (m^0 \oplus h(r))||r.$$

We find the ciphertext by squaring the padding message

$$c = y^2 \mod N.$$

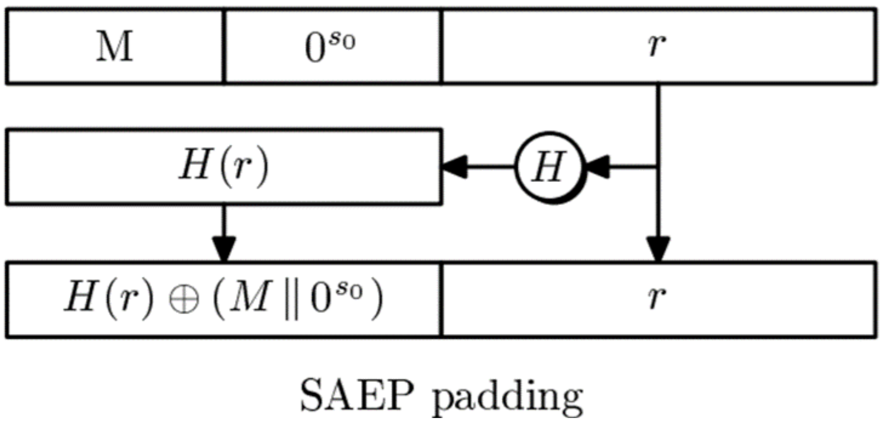

Figure 1: A schematic representation of the SAEP protocol.

For the working of the decryption and for the proof we constrain the lengths of the padded message. We say $l \leq \frac{k}{4}$ and $l + s_0 \leq \frac{k}{2}$ and $k = l + l_0 + l_1$. Remark that y is now a k bit-string, hence is an integer with $2^{k-1} \leq y < 2^{k+1}$.
The last conclusion is important for the decryption. The integers y and N have length k- and respectively $k+2$-bits. If we calculate the roots of c, we delete the roots that are bigger than $\frac{N}{2}$. Then two or one root remain that are possibilities for y.
If two roots remain, we call these y_1 and y_2. Working the protocol backwards, we find $y_1 = v_1||r_1$ and $y_2 = v_2||r_2$. From here we can compute $m_1^0 = v_1 \oplus r_1$ and $m_2^0 = v_2 \oplus r_2$. For both, we check if the last l_0-bits are zeros. The one where this is the case, that is the original message. There is a small change, this is the case for both m_1^0 and m_2^0. But this occurs with an ignorable probability.

8.2 The Coppersmith Theorem

Before we give the security claim, we state the Coppersmith theorem without proof. This theorem is an important tool for the proof of the security of *SAEP*-Rabin. [7, Corollary 1] [8, Theorem 9.18]

Theorem 8.1 (Coppersmith)**.** *Let $N \in \mathbb{N}$, and the function $f : \mathbb{Z}_N \to \mathbb{Z}_N$ a polynomial function of degree d. Then there is an efficient algorithm which finds in time $O(\frac{1}{d}\log(N))$ all $x \in \mathbb{Z}$ such that $f(x) = 0 \mod N$ and $|x| < N^{\frac{1}{2}}$.*

From this theorem the next corollary follows:

Corollary 8.1. *Let $N, c \in \mathbb{N}$ and $f(x) : \mathbb{Z}_N \to \mathbb{Z}_N, x \mapsto x^2 - c$. Then there is an efficient algorithm which finds solutions in $O(\frac{1}{2}\log(N))$ time for $|x| < N^{\frac{1}{2}}$.*

Remark that, this corollary we cannot apply to the Rabin primitive. For the function $f(x) = x^2 - c$ we find only roots y smaller than $N^{\frac{1}{2}} < 2^{\frac{k+1}{2}}$. Because the roots y has to have length k we know $2^{k-1} \leq y < 2^k$.
To relate the Coppersmith Theorem to the *SAEP*-Rabin we state the following lemma. [7]

Lemma 8.1. *Let $y = m||t||r$ be a bit-string an unknown root of c in $\mathbb{Z}_N$ of bit length k, and the lengths of m, t and r are respectively l, l_0 and l_1 such that $l \leq \frac{k}{4}$ and $l + l_0 \leq \frac{k}{2}$. Then:*

1. *If r is given then y can be calculated in $O(\frac{1}{2}\log(N))$ time.*
2. *If m is given and another ciphertext c' such that the last $l_0 + l_1$ bits of y' are equal to y, then y can be calculated in $O(\frac{1}{4}\log(N))$ time.*

Proof. We want to find solutions y for $y^2 = c \mod N$. In this proof we treat both conclusions separately.

1. Assume we know r. We want to calculate $v = m||t$. Following the assumption, we get $|v| < 2^{l+l_0} \leq 2^{\frac{k}{2}} < N^{\frac{1}{2}}$. Another remark we need to make is $v||0^{s_1} = 2^{s_1} \cdot v$.
   We consider the polynomial $f(x) = (2^{s_1}x + r)^2 - c \mod N$. This satisfied the Coppersmith Theorem and we calculate v and y in $O(\frac{1}{2}\log(N))$ time.

2. Assume we know m. We want to calculate $w = t||r$. Further we assume we have an another element $c' \in \mathbb{Z}_N$ such that the $y'^2 = c' \mod N$ with $y' = m'||w$. Now we compare y and y' to get an polynomial function for which we can apply the Coppersmith Theorem.
   Because only the first bits are different, we can write $y' = y + 2^{s_0+s_1}\delta$, with $\delta = m - m'$. Notice that $|\delta| < 2^l \leq 2^{\frac{k}{4}} < n^{\frac{1}{4}}$. To solve δ, we have to find the roots of the function $f(x) = X^2 - c$ and $g(X, \delta) = (X + 2^{s_0+s_1}\delta)^2 - c'$. Referring to [8, p. 309] we state that this gives an polynomial of degree 4. Here we can apply Coppersmith's Theorem and find w and y in $O(\frac{1}{4}\log(N))$ time.

□

8.3 Security claim

Now we have stated the Coppersmith Theorem and Lemma 8.1 we can give and prove the security claim. We follow the proof as described in [8, Section 9.1.5]

Theorem. *Assume the h is ideal hash-function. Assuming adversary A breaks SAEP-Rabin with running time t and probability of success greater than $1/2+\epsilon$, and q_d resp. q_h the number of decryption and hashes queried by A,*
then there is an algorithm B for factoring the modulus N with execution time $t + O(\frac{1}{2}q_h q_d \log(N) + \frac{1}{4}q_d\log(N))$ and chance of success greater than $\frac{1}{6} \cdot \epsilon \cdot (1 - \frac{q_d}{2^{s_0}} - 2\frac{q_d}{2^{s_1}})$.

Proof. In this proof we construct an algorithm B that can factor the modulus N into the primes p and q. We will design an algorithm B_1 in the next part of the proof that can find roots of an integer c with $c = y^2 \mod N$. We will show that the probability of success for B_1 to find a root y is bigger than $\epsilon \cdot (1 - \frac{q_d}{2^{s_0}} - 2\frac{q_d}{2^{s_1}}$ and the execution time is $t + O(\frac{1}{2}q_h q_d \log(N) + \frac{1}{4}q_d \log(N)$.
If B_1 finds a root $a' \neq a$, then we can easily find one of the prime dividers of N by calculating $\gcd(a' - a, N)$. The probability for this condition is minimized by $\frac{1}{6}$. [8]. We conclude that B factors N with probability of success greater than $\frac{1}{6} \cdot \epsilon \cdot (1 - \frac{q_d}{2^{s_0}} - 2\frac{q_d}{2^{s_1}})$.

Algorithm B_1

We will give the design of B_1. The adversary A is interacting with B_1. We need the datasets S for the queries of r and $h(r)$ that A made. We write the cardinality of this set as $|S| = q_d$. There are three options that A can query for B_1:

1. The adversary queries for a hash-value of r. Then
   (a) if $r \in S$ then reply with $h(r)$ to B_1,
   (b) else B_1 tries to find a root y of c following point 1 of Lemma 8.1.
   (c) If this is successful, then B_1 has done his task. Else B_1 saves r and a random chosen $h(r)$ in S.
2. The adversary queries for a ciphertext c'. Then
   (a) B_1 try to find a root y from point 2 of Lemma 8.1.
   (b) If this is not successful then B_1 tries to find to find roots for c' for all $r' \in S$ with point 1 of Lemma 8.1. If B_1 successful find a root and the belonging message m' is valid, then it return m'.
   (c) Else B_1 returns that c' is an invalid ciphertext.
3. The last option is when A give message m_1 and m_2 and S give a random encryption c.

Analysis of B_1

In this section of the proof we analyze algorithm B_1 and give the change of success to find roots $y_{1,2}$ of c. First we give two cases in which B_1 successfully finds y_1 or y_2 with $y_i = v_i||r_i$, $i = \{0, 1\}$.

1. The algorithm B_1 find y_1 or y_2 if r_1 or r_2 is queried in step $1(b)$ of the algorithm.
2. The algorithm B_1 find y_1 or y_2 if a ciphertext c' such that the $l_0 + l_1$ bits of the corresponding roots are the same as y_1, y_2, by step $2(a)$ of the algorithm.

Thirdly we have the case that step $2(b)$ is successful in finding a message:

3. The algorithm B_1 is successful in finding a message if A queries a ciphertext c' and if A has previously asked for $h(r)$.

Now we look at two options where B_1 is not successful and fails in to find proper roots.

4. The first difference between B_1 and Alice is that B_1 has to restrict to the values that are saved. Hence, if one of the two messages m_0, m_1 is encrypted with ciphertext c it is possible that B_1 does not find the message.
   It is possible that B_1 finds one of the roots, but this give no proper message: $h(r_i) \neq (m_b)||0^{l_0}$ for $b = 0, 1$. Hence, we did not queried the right r_i. The probability that we queried r_1 or r_2 is $\leq 2\frac{q_d}{2^{l_1}}$.
5. The last difference between B_1 and Alice is that all previous cases are valid, but B_1 rejects a ciphertext c'. This is only possible if $h(r')$ is never queried before. The probability for the case that $h(r')$ is queried is $\leq 2\frac{q_d}{2^{l_0}}$.

The adversary A has success chance of $\frac{1}{2}+\epsilon$ to break $SAEP$-Rabin. If A does not have the data S this probability is $\frac{1}{2}$. Hence, the probability to find a root in step 1 and 2 is ϵ. As we argue the change of cases 4 and 5 are smaller than $2\frac{q_d}{2^{l_1}}+2\frac{q_d}{2^{l_0}}$. Hence the probability of success of algorithm B_1 is greater than $\epsilon\cdot(1-2\frac{q_d}{2^{l_1}}+2\frac{q_d}{2^{l_0}})$.
Finally we analyse the runtime of B_1. This is the summation of the time that A needs and the steps as described in the previous paragraph. The first is given as time t. The last is bounded by $O(\frac{1}{2}q_dq_h\log(N)+\frac{1}{4}q_d\log(N)$. Hence total the running time of B is

$$t+O(\frac{1}{2}q_dq_h\log(N)+\frac{1}{4}q_d\log(N)).$$

$\square$

9 Conclusion

In this section we want to compare the result of the security proof as described in the previous chapters. We discuss the assumptions behind the theorems and the difference between them. In [9, Section 4] Koblitz and Menezes give two main points where $SAEP+$-Rabin has advantage to $OAEP+$-RSA.

9.1 Factoring or inverting

The first difference we discuss between the security of $OAEP+$-RSA and $SAEP$-Rabin is the mathematical problem in the reduction argument.
The security claim of $OAEP+$-RSA reduces to finding the inverse of the RSA-function. In [3] Boneh describe an overview of possible attacks on the RSA primitive. Most of the attacks can be prevented by customizing the protocol. Requirements can be set on the private and public keys. For example an attack described by M. Wiener gives restrictions on the private key d. Wiener proved that if $N = p \cdot q$ is the modulus of the RSA function and the private exponent is $d < \frac{1}{3}N^{\frac{1}{4}}$ then an adversary Charly can find d efficiently.
If the receiver has to take the restrictions in account, the RSA-function is a secure primitive to use. The assumption that there is no efficient method to find the inverse is still valid.
The security claim of $SAEP+$-Rabin reduces to the factoring of integers. History shows that mathematicians not find an efficient algorithm to find prime divisors of an integer. Theoretically such an algorithm can exist, but cryptographers assume it will not be found soon.
The assumption of factoring appears better than the inverting assumption. So $SAEP+$-Rabin is in favor to choose in cryptographic above $OAEP+$-RSA. [8, Section 3.3] [9]
However, in the field of 'provable security', one of the most important questions is whether factoring reduces to finding the inverse of RSA. The quest for an answer is illustrated by papers with titles like 'Breaking RSA may not be equivalent to factoring' [5] and 'Breaking RSA May Be As Difficult As Factoring' [6]. [10, Section 2]

9.2 Reduction tightness

A second difference between the security proofs is the tightness of the reduction. The reduction gap of $OAEP+$-RSA is $O(q_G q_{H_1} k^2 + (q_G + q_{H_2} + q_{H_1} + q_d)k)$. We choose the security parameter k approximately equal to N, we can estimate the reduction gap $O(N^2)$. If we compare this to the reduction gap $O(\log N)$ of $SAEP$-Rabin, we see that the last one has a tight reduction. Considering this difference $SAEP$-Rabin is again preferably to use.
The reduction of $OAEP+$-RSA is not a tight reduction. How can we respond on this? Koblitz and Menezes describe several reactions to a non-tight reduction. These vary from "a non-tight reduction is better than noting at all" and "the protocol is secure in practice, even a tight reduction may simply not exist" to "the protocol is in fact insecure, but an attack has not yet been discovered." [10, Section 4]
In my opinion a non-tight reduction for the RSA protocol does not imply that it is insecure. During the implementation of the protocol one has to be aware of the non-tightness of the reduction and must choose a larger modulus N.
A further remark is that the security proof of $OAEP+$-RSA in chapter 7 was for all exponents e. In [14, Section 7.2], Shoup argued that for $e = 3$ there exist a tight reduction. In the proof he used theorem 8.1 of Coppersmith.

9.3 Discussion

After the comparison of two important points we can conclude that $SAEP$-Rabin is the best choice. Koblitz and Menezes summarize this conclusion "Boneh shows that it is actually much better to apply Rabin encryption (...) rather than the RSA function, for two reasons: (1) the assumption that finding eth roots modulo n is hard is replaced by the weaker and more natural assumption that factoring is hard (...); and (2) the reduction argument is tight." [9, Page 9]
Given this conclusion, it is remarkable to see that in practice Rabin is rarely used. Cryptographers distrust the Rabin primitive. The reason for this distrust is that for the Rabin encryption they found

the first tight-reduction argument for security claim ever that however, failed in a Chosen Ciphertext Attack. For $SAEP$-Rabin this suspicion is unfounded, because a stage of padding is added.

9.4 Provable security

In 1994, Bellare and Rogaway introduced the $OAEP$ padding protocol. [2] Seven years later Shoup found a fallacy in the security proof of the protocol and he proposed the $OAEP+$ padding protocol. [14] This shows that there is still much to be gained for the field of 'provable security'. The question was raised how serious security proofs are taken. The reduction argument must be reviewed by other mathematicians to find that the proof is correct. [9, Page 39]
As nowadays the overwhelmingly majority of our communication is over the internet, we need to find the best way to protect the content of our communication. Mathematicians play an important role in searching for encryption primitives and security proofs. This important role in an important topic calls for accessibility, comprehensibility and reliability of 'provable security'. The mathematicians Koblitz and Menezes started in 2006 with a series of papers to work towards this goal[2]. This thesis arose from diving in the material of some of these papers.

[2]On the website http://anotherlook.ca/ the papers of Koblitz and Menezes are collected.

References

[1] Mihir Bellare. Practice-oriented provable-security. In *International Workshop on Information Security*, pages 221–231. Springer, 1997.

[2] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 92–111. Springer, 1994.

[3] Dan Boneh. Twenty years of attacks on the f cryptosystem. *Notices-American Mathematical Society*, 46:203–213, 1999.

[4] Dan Boneh. Simplified OAEP for the RSA and Rabin functions. In *Annual International Cryptology Conference*, pages 275–291. Springer, 2001.

[5] Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 59–71. Springer, 1998.

[6] Daniel RL Brown. Breaking RSA may be as difficult as factoring. *IACR Cryptology ePrint Archive*, 2005:380, 2005.

[7] Don Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *Journal of Cryptology*, 10(4):233–260, 1997.

[8] Hans Delfs, Helmut Knebl, and Helmut Knebl. *Introduction to cryptography*, volume 3. Springer, 2015.

[9] Neal Koblitz and Alfred Menezes. Another look at "provable security". *Journal of Cryptology*, 20(1):3–37, 2006.

[10] Neal Koblitz and Alfred Menezes. Another look at provable security. II. In *International Conference on Cryptology in India*, pages 148–175. Springer, 2006.

[11] Neal Koblitz and Alfred Menezes. Another look at security definitions. *Advances in Mathematics of Communications*, 7(1):1–38, 2013.

[12] Michael O Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, Massachusetts Inst. of Tech. Cambridge Lab for Comp. Science, 1979.

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

[14] Victor Shoup. OAEP reconsidered. In *Annual International Cryptology Conference*, pages 239–259. Springer, 2001.

[15] Gerard Tel. *Cryptografie, beveiliging van de digitale maatschappij.* Instituut voor Informatica en Informatiekunde Universiteit Utrecht, 2006.