An exotic fieldtrip through cryptography

The extension of RSA encryption

W.H. van Barneveld

TUDelft

An exotic fieldtrip through cryptography

The extension of RSA encryption

by

W.H. van Barneveld

to obtain the degree of Bachelor of Science
at the Delft University of Technology,

Student number:	5309905	
Project duration:	April, 2023 – July, 2023	
Thesis committee:	Dr. N.D. Verhulst,	TU Delft, supervisor
	Dr. J.A.M. de Groot,	TU Delft

TU Delft

Preface

This thesis is written to finish the bachelor Applied Mathematics and obtain the degree of Bachelor of Science at the Delft University of Technology. Last autumn semester I took a course in cryptography. This course sparked my interest in this branch of mathematics. Therefore I was pleased when I got the confirmation that I got an assignment about cryptography assigned for my bachelor thesis.

I want to thank my supervisor, Dr. N.D. Verhulst, for guiding me these past months. Furthermore I would like to thank Dr. J.A.M. de Groot for being part of my Bachelor Committee. Last but not least I would like to thank my family, friends and fellow students for their support.

W.H. van
Barneveld
Delft, July 2023

Contents

1 Introduction

In 1971 the first email has been sent. Since then this and more advanced digital technology has become part of our day-to-day life. The rise of digital communication also brought the necessity of online security and digital signatures. Where before the presence of a seal made it clear whether or not a letter from one person to another has been read by some eavesdropper a new method had to be implicated to get something similar to send digital letters. One of the first systems developed to provide online security is what is now known as RSA encryption. This method has been first published by Rivest, Shamir and Adleman in 1978 [10]. Even on this day, a modification of the first method of RSA encryption is being used and considered safe. However no proof exists that an efficient method to break the security of the RSA encryption can not be developed in the future.

At the end of last year, Chinese researchers have made the claim that they have figured out a method to break RSA using quantum computers [13]. We could increase the length of the keys since they claim to be able to break RSA with keys of 2048 bits (approximately 600 digits) or less. The idea that the development of quantum computers would impose great risks on our data is not new. In 1994 Peter Shor already published a possible algorithm that would make it possible to break the most important part for the security of RSA encryption [11]. However, in 1994 this could only exist in theory since the necessary quantum computer did not exist.

If the day comes that RSA encryption is not considered safe anymore we need to implement a new form of security. Many scientists have been working on new cryptosystems unrelated to RSA-encryption. However, there is also the possibility that we could increase the key length or modify RSA encryption in such a way that it is secure again, or so we have some more time to develop the new cryptosystem without leaking all information that is currently secured with RSA encryption.

Instead of increasing the key length, we also might be able to extend the RSA algorithm to different mathematical structures, where the security still might hold. This paper explores a number of these structures and describes whether or not it could be possible to extend RSA encryption. In this paper we will focus on verifying whether the RSA encryption and decryption algorithms are well defined on the mathematical structures.

In Chapter 2 the mathematics necessary to understand the algorithm that will be introduced. Chapter 3 explains the RSA encryption, talks about the underlying mathematics and the security of this system and will explain which part of RSA encryption will be focused on. Chapter 4 explores the implementation of RSA encryption on different classes of semifields.

2 Preliminaries

The RSA encryption method relies on branches of mathematics such as algebra and number theory. In order to understand the mathematics behind the RSA algorithm for both the encryption and decryption, some mathematical concepts will be introduced.

2.1. Algebra

Theorem 2.1.1 (Bezout's Identity). *Let $a, b \in \mathbb{Z}$ such that $a, b \neq 0$, and $d = gcd(a, b)$. Then $x, y \in \mathbb{Z}$ exists such that $xa + yb = d$.*

Proof. Cf. e.g. [5] □

The next corollary is a special case of Bezout's Identity.

Corollary 2.1.2. *$a, b \in \mathbb{Z}$ unequal to zero are coprime ($gcd(a, b) = 1$) if and only if there exist $x, y \in \mathbb{Z}$ such that $xa + yb = 1$.*

For a and b integers, the Euclidean Algorithm can be used to d such that $d = gcd(a, b)$. In chapter 3 we will see that this algorithm, with minor modifications, can also be used to find the values for x and y such that $xa + yb = d$ with $gcd(a, b) = d$.

Algorithm 2.1.3 (The Euclidean Algorithm). *Let a and b be integers such that $a \geq b \geq 0$. A sequence $r_0, r_1, \ldots$ of (non negative) integers is going to be constructed. Start by setting $r_0 = a$ and $r_1 = b$. Assume r_n is the last computed term. Define*

$$r_{n+1} = r_{n-1} - q_n \cdot r_n$$

where q_n is the largest integer such that $0 \leq r_{n+1} < r_n$. In other words; r_{n+1} is the remainder when dividing r_{n-1} by r_n. Continue this process until $r_n = 0$. Now return $gcd(a, b) = r_{n-1}$.

Example 2.1.4 (Euclidean Algorithm). *The Euclidean Algorithm will be used to find $gcd(254, 32)$. Start by setting $r_0 = 254$ and $r_1 = 32$. Now $r_2 = 254 - q_1 \cdot 32$. Since q_1 must be the largest integer such that $0 \leq r_2 < 32$, we find $q_1 = 7$ which gives $r_2 = 30$. Repeat this to find $q_2 = 1$ and $r_3 = 2$. Repeating again gives $q_3 = 15$ and $r_4 = 0$. Since $r_4 = 0$ we return r_3, that is gcd$(254, 32) = 2$.*

Definition 2.1.5 (Group). *A set G with an operation $\circ$, $G \times G \rightarrow G$, is a group when the following requirements are met.*

G1 (Associativity) $a \circ (b \circ c) = (a \circ b) \circ c$ for all $a, b, c \in G$.

G2 (Identity element) There exists an $e \in G$ such that $e \circ a = a \circ e = a$ for all $a \in G$.

G3 (Inverse element) For all $a \in G$, there exists an $a^* \in G$ such that $a \circ a^* = a^* \circ a = e$.

Definition 2.1.6. *Let G be a group and let S be a subset of G. Define $H \subseteq G$ as the set of those elements $g \in G$ that can be written as $g = s_1 s_2 \cdots s_n$ with $s_i \in S$ or $s_i^{-1} \in S$ for all $i = 1, \dots, n$. Then H is a subgroup of group G.*

Definition 2.1.7. *Let G_1 and G_2 be groups. A map $f : G_1 \to G_2$ is a group homomorphism if for all $a, b \in G_1$*

$$f(ab) = f(a)f(b)$$

where multiplication of a and b on the left-hand side takes place in group G_1, and the multiplication of $f(a)$ and $f(b)$ on the right-hand side takes place in the group G_2. A bijective homomorphism is an isomorphism. If such isomorphism exists we say that G_1 and G_2 are isomorphic.

Theorem 2.1.8. *Let G_1 and G_2 be groups with identity elements e_1 and e_2 respectively. Let $f : G_1 \to G_2$ be a group homomorphism. Then $f(e_1) = e_2$ and $f(x^{-1}) = f(x)^{-1}$ for all $x \in G_1$.*

Proof. Cf. e.g. [3] p.38. □

Definition 2.1.9. *Let G be a group. A map $f : G \to G$ is an automorphism if it is an isomorphism from G to itself. That is f is bijective and preserves the structure of the group G, i.e. for $\circ$ an operation on group G and $a, b \in G$, $f(a \circ b) = f(a) \circ f(b)$.*

Definition 2.1.10. *Let G_1 and G_2 be groups with identity elements e_1 and e_2 respectively. Let $f : G_1 \to G_2$ be a group homomorphism. The kernel of f is defined as $ker(f) = \{x \in_1 : f(x) = e_2\}$.*

Theorem 2.1.11. *Let G_1 and G_2 be groups with identity elements e_1 and e_2 respectively. Let $f : G_1 \to G_2$ be a group homomorphism. f is an injective group homomorphism if and only if $ker(f) = e_1$.*

Proof. Assume f is injective. For $x \in ker(f)$, $f(x) = e_2 = f(e_1)$ by Theorem 2.1.8. Since f is injective we must have $x = e_1$ and therefore $ker(f) = e_1$. Assume $ker(f) = e_1$. Suppose $f(x) = f(y)$ for some $x, y \in G_1$. Then by Theorem 2.1.8, $f(xy^{-1}) = f(x)f(y)^{-1} = e_2$. This would imply $xy^{-1} = e_1$, hence $x = y$. Therefore f is injective.

□

A well-known example of a group is a multiplicative group $(\mathbb{Z}/n\mathbb{Z})^*$.

Definition 2.1.12. *Define a multiplicative group by*

$$(\mathbb{Z}/n\mathbb{Z})^* = \{\overline{a} \in \mathbb{Z}/n\mathbb{Z} : gcd(a, n) = 1\}.$$

Here, $\overline{a} \in \mathbb{Z}/n\mathbb{Z}$ means $\overline{a} \equiv a \ (mod\ n)$, that is the remainder when dividing a by n. For example, if $n = 6$, then $(\mathbb{Z}/6\mathbb{Z})^* = \{1, 5\}$. For p prime, $(\mathbb{Z}/p\mathbb{Z})^* = \{\overline{1}, \overline{2}, \dots, \overline{p-1}\}$ so if $n = 7$ one has $(\mathbb{Z}/7\mathbb{Z})^* = \{1, 2, 3, 4, 5, 6\}$.

Euler's totient function counts the number of (positive) integers up to n that are coprime to n. Therefore it also counts the number of elements in a multiplicative group.

Definition 2.1.13 (Euler's totient function)**.**

$$\phi(n) = \#\{a \in \mathbb{Z} : 1 \leq a \leq n,\ gcd(a, n) = 1\} = \#(\mathbb{Z}/n\mathbb{Z})^*.$$

Since a prime number is coprime with all (positive) integers smaller that itself we have for p prime that $\phi(p) = p - 1$.

Definition 2.1.14 (Order)**.** *Let G be a finite group with identity element e. If n is the smallest positive integer such that $x^n = e$ for $x \in G$, we say that the order of x is n, and use notation $ord(x) = n$. The order of a group G is the number of elements of G denoted by either $\#G$ or $ord(G)$.*

As a result of the definitions of Euler's totient function and the order of a group (Definition 2.1.13 and Definition 2.1.14), for the multiplicative group we have $ord((\mathbb{Z}/n\mathbb{Z})^*) = \phi(n)$.

The next corollary will give a relation between the order of an element and the order of the group the element belongs to.

Corollary 2.1.15. *Let G be a group with finitely many elements. Then for all $x \in G$, $ord(x)|ord(G)$.*

Proof. Cf. e.g. [3]. □

Similarly, there is a relation between the order of a group G and the order of a subgroup H of G. The next corollary is an result of Lagrange's Theorem which will not be discusses here. For Lagrange's Theorem we refer to [3] p.59-64.

Corollary 2.1.16. *Let G be a group and $H \subseteq G$ a subgroup of G. Then $ord(H)|ord(G)$.*

Euler's product formula can be used to easily compute Euler's totient function, and thus determines the order of the multiplicative group.

Theorem 2.1.17 (Euler's product formula)**.** *Let $n \in \mathbb{Z}$ with $n > 0$. Then*

$$\phi(n) = n \cdot \prod_{p|n,\ p\ prime} \left(1 - \frac{1}{p}\right).$$

Proof. Cf. e.g. [12] p.271-272. □

Whenever $n = p$ where p is a prime, we get $\phi(p) = p \cdot \prod_{q|p,\ q\ prime} \left(1 - \frac{1}{q}\right) = p - 1$.

Corollary 2.1.18 (Euler's Theorem)**.** *Let $a, m \in \mathbb{Z}$ be such that $m > 0$ and $gcd(a, m) = 1$. Then*

$$a^{\phi(m)} \equiv 1\ (mod\ m),$$

Proof. $gcd(a, m) = 1$ implies $\overline{a} \in (\mathbb{Z}/m\mathbb{Z})^*$. Therefore $ord(\overline{a})|ord((\mathbb{Z}/m\mathbb{Z})^*) = \phi(m)$. As a result $\overline{a}^{\phi(m)} = \overline{1}$. Which is $a^{\phi(m)} \equiv 1\ (mod\ m)$. □

2.2. Fermat's Little Theorem

Theorem 2.2.1 (Fermat's Little Theorem)**.** *Let p be a prime number and let $a \in \mathbb{Z}$. Then*

$$a^p \equiv a\ (mod\ p).$$

Proof. If $p|a$, then $a^p \equiv 0 \equiv a\ (mod\ p)$. Assume $p \nmid a$. Since p is prime, $gcd(a, p) = 1$ and $\phi(p) = p-1$. Together with Euler's Theorem, Corollary 2.1.18 we have $a^{\phi(p)} \equiv a^{p-1} \equiv 1\ (mod\ p)$. Multiply both sides by a to get $a^p \equiv a\ (mod\ p)$

□

Besides the proof stated above, many methods are known to prove Fermat's Little Theorem such as using induction, binomial expansion and using Lagrange's Theorem. Above a proof for Fermat's Little Theorem using Euler's Theorem is given. Another proof for Fermat's Little Theorem, this time without Euler's Theorem will be given.[1]

Proof. Consider $(\mathbb{Z}/p\mathbb{Z})^* = \{1, 2, ..., p-1\}$. Define the set $S = \{\overline{a}, \overline{2a}, ..., \overline{(p-1)a}\}$, which we get by multiplying all the elements in $(\mathbb{Z}/p\mathbb{Z})^*$ with some integer $a \in \mathbb{Z}$ such that $p \nmid a$ and take them modulo p. By the coprimality of a and p, all elements in S are distinct. Hence $\#S = \#(\mathbb{Z}/p\mathbb{Z})^* = p-1$ by Euler's product formula, Theorem 2.1.17. Moreover, since all elements in S are taken modulo p and $p \nmid a$, $0 < s < p$ for all $s \in S$. Together these imply $S = (\mathbb{Z}/p\mathbb{Z})^*$. Therefore

$$\prod_{s \in S} s \equiv \prod_{q \in (\mathbb{Z}/p\mathbb{Z})^*} q\ (mod\ p).$$

By the commutative property of multiplication

$$a^{p-1}(p-1)! \equiv (p-1)!\ (mod\ p).$$

By the cancellation law

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

Multiply both sides by a to derive

$$a^p \equiv a\ (mod\ p).$$

We are left with the case when $p \mid a$. Then both $a^p \equiv 0\ (mod\ p)$ and $a \equiv 0\ (mod\ p)$, hence $a^p \equiv a\ (mod\ p)$. □

2.3. Finite fields

Before one is able to define finite fields, it is necessary to know the definition of a field. A field is a mathematical structure in which addition, subtraction, multiplication and division (excluding division by zero) can be performed. For these operations the field axioms should hold. Examples of fields are the rational numbers $\mathbb{Q}$ and the real numbers $\mathbb{R}$.

Definition 2.3.1 (Field axioms)**.** *Let F be a set with operations $+$ and $\cdot$, $(F, +, \cdot)$ is a field if and only if the following properties are satisfied if for all $a, b, c \in F$:*

- *A1 (Associativity) $(a + b) + c = a + (b + c)$*
- *A2 (Commutativity) $a + b = b + a$*
- *A3 (Additive identity) there exists a $0 \in F$ such that $a + 0 = a$*
- *A4 (Additive inverse) there exists a $-a \in F$ such that $a + (-a) = 0$*
- *DL (Distributivity) $a(b + c) = ab + ac$*
- *M1 (Associativity) $(ab)c = a(bc)$*
- *M2 (Commutativity) $a \cdot b = b \cdot a$*
- *M3 (Multiplicative identity) there exists an $1 \in F$ such that $a \cdot 1 = a$*
- *M4 (Multiplicative inverse) if $a \neq 0$, there exists an $a^{-1} \in F$ such that $a \cdot a^{-1} = 1$*
- *The additive identity 0 and the multiplicative identity 1 are different, that is $0 \neq 1$.*

Definition 2.3.2 (Field characteristic)**.** *The field characteristic of a field F with unit element e is the smallest $n \in \mathbb{N}$ such that*

$$0 = ne = \underbrace{e + e + \ldots + e}_{n \text{ summands}}.$$

We denote the field characteristic of field F by $char(F) = n$. If no such n exists we say $char(F) = 0$.

Theorem 2.3.3. *The field characteristic of any field is either a prime or is equal to zero.*

Proof. Let F be a field with field characteristic $char(F) = m$ with m not a prime. Since m is not a prime, there exist integers r and s smaller than m such that both r and s divide m and in particular $m = r \cdot s$. By the definition of the field characteristic, $0 = me = (re)(se)$. This implies either $re = 0$ or $se = 0$, but since r and s are smaller than m, this contradicts m being the smallest integer such that $me = 0$. □

A finite field F is a finite set where the field axioms hold. Per definition, this set F is closed under both addition and multiplication. i.e. for all $a, b \in F$, $a + b \in F$ and $a \cdot b \in F$. Finite fields are also called Galois fields after the founder of finite field theory, Évariste Galois.

Theorem 2.3.4. *The field order, i.e. the number of elements of a finite field, is always a prime or the power of a prime.*

Proof. Before, in Theorem 2.3.3, we have shown that every finite field has a prime number p as the field characteristic. Therefore every field F has a subfield H, where H is isomorphic to $\mathbb{Z}_p = \{0, 1, \ldots, p - 1\}$. By Corollary 2.1.16, $ord(H) | ord(F)$. Hence $ord(F) = k \cdot ord(H)$ for some integer k. Since a field is a vector space over any subfield, $ord(F) = (ord(H))^l$ for some integer l. □

Up to isomorphisms, these finite fields are unique.

A finite field with p^n elements, where p is prime and n a natural number, is often denoted by $GF(p^n)$. For $n = 1$, the integers modulo p form a field with field order p, thus $GF(p) = \{0, 1, \ldots, p - 1\}$. All operations are performed modulo p. The field $GF(p^n)$ has field characteristic p.

When $n > 1$, then the elements of $GF(p^n)$ are polynomials with degree up to $n - 1$ instead of 'just' integers. The coefficients for the polynomials are elements of $GF(p)$. Addition is as expected; $\sum_{i=0}^{n-1} a_i x^i + \sum_{i=0}^{n-1} b_i x^i \equiv \sum_{i=0}^{n-1} c_i x^i$, where $c_i \equiv a_i + b_i \ (mod\ p)$ for all $\sum_{i=0}^{n-1} a_i x^i, \sum_{i=0}^{n-1} b_i x^i \in GF(p^n)$. To define multiplication over $GF(p^n)$ we need to decide on an irreducible polynomial (over the integers) $g(x)$ of degree n. $g(x)$ being an irreducible polynomial means that the coefficients of g are elements of $GF(p)$ but $g(x)$ can not be factored in the product of two polynomials (both non-constant). Multiplication is defined as the remainder of the product after dividing by the polynomial $g(x)$. Subtraction is defined similar as addition. Division is the same as multiplication by the inverse element, which exists by axiom $M4$. Different choices of irreducible polynomials yield different but isomorphic (finite) fields. Some examples of finite fields with corresponding addition and multiplication tables will be given.

The elements of $GF(p^n)$ can be written using multiple representations. In this paper the polynomial representation will be used. Other representations are the representation by powers of x or using binary. Some examples of finite fields will be discussed.

Example 2.3.5 ($GF(4)$)**.** *Since $4 = 2^2$, an irreducible polynomial of degree 2 will be used in the construction of this finite field. Take $g(x) = x^2 + x + 1$ as the irreducible polynomial in $GF(4)$. The elements of this finite field are $GF(4) = \{0, 1, x, x + 1\}$. Multiplication and addition are as in the addition and multiplication table below. Addition is performed modulo 2. As a result of our decision on the irreducible polynomial, $x^2 \equiv x + 1$, this is used in constructing the multiplication table.*

Addition Table

$+$	0	1	x	$x+1$
0	0	1	x	$x+1$
1	1	0	$x+1$	x
x	x	$x+1$	0	1
$x+1$	$x+1$	x	1	0

Multiplication Table

$\cdot$	0	1	x	$x+1$
0	0	0	0	0
1	0	1	x	$x+1$
x	0	x	$x+1$	1
$x+1$	0	$x+1$	1	x

Example 2.3.6 ($GF(9)$)**.** *Since $9 = 3^2$ again an irreducible polynomial of degree 2 will be used in the construction of this finite field. Take $g(x) = x^2 + 1$ as the irreducible polynomial in $GF(3)$. Instead of working modulo 2 we will work modulo 3. The elements of this finite field are $GF(9) = \{0, 1, 2, x, x + 1, x + 2, 2x, 2x + 1, 2x + 2\}$. Multiplication and addition are as in the addition and multiplication table. Addition is performed modulo 3. As a result of our decision on the irreducible polynomial, $x^2 = -1 \equiv 2$, this is used in constructing the multiplication table.*

Addition Table

$+$	0	1	2	x	$x+1$	$x+2$	$2x$	$2x+1$	$2x+2$
0	0	1	2	x	$x+1$	$x+2$	$2x$	$2x+1$	$2x+2$
1	1	2	0	$x+1$	$x+2$	x	$2x+1$	$2x+2$	$2x$
2	2	0	1	$x+2$	x	$x+1$	$2x+2$	$2x$	$2x+1$
x	x	$x+1$	$x+2$	$2x$	$2x+1$	$2x+2$	0	1	2
$x+1$	$x+1$	$x+2$	x	$2x+1$	$2x+2$	$2x$	1	2	0
$x+2$	$x+2$	x	$x+1$	$2x+2$	$2x$	$2x+1$	2	0	1
$2x$	$2x$	$2x+1$	$2x+2$	0	1	2	x	$x+1$	$x+2$
$2x+1$	$2x+1$	$2x+2$	$2x$	1	2	0	$x+1$	$x+2$	x
$2x+2$	$2x+2$	$2x$	$2x+1$	2	0	1	$x+2$	x	$x+1$

Multiplication Table

$\cdot$	0	1	2	x	$x+1$	$x+2$	$2x$	$2x+1$	$2x+2$
0	0	0	0	0	0	0	0	0	0
1	0	1	2	x	$x+1$	$x+2$	$2x$	$2x+1$	$2x+2$
2	0	2	1	$2x$	$2x+2$	$2x+1$	x	$x+2$	$x+1$
x	0	x	$2x$	2	$x+2$	$2x+2$	1	$x+1$	$2x+1$
$x+1$	0	$x+1$	$2x+2$	$x+2$	$2x$	1	$2x+1$	2	x
$x+2$	0	$x+2$	$2x+2$	$2x+2$	1	x	$x+1$	$2x$	2
$2x$	0	$2x$	x	1	$2x+1$	$x+1$	2	$2x+2$	$x+2$
$2x+1$	0	$2x+1$	$x+2$	$x+1$	2	$2x$	$2x+2$	x	1
$2x+2$	0	$2x+2$	$x+1$	$2x+1$	x	2	$x+2$	1	$2x$

Example 2.3.7 ($GF(8)$)**.** *Instead of an irreducible polynomial of degree 2, for the construction of the finite field $GF(8)$ we will need an irreducible polynomial in $GF(2)$ of degree 3. The polynomial $g(x) =$*

$x^3 + x + 1$ will be used. Since this time we have an irreducible polynomial of degree 3, the elements of our finite field also include polynomials of degree 2. The full set of elements is $GF(8) = \{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$. Multiplication and addition are as in the addition and multiplication table. Addition is performed modulo 2. As a result of our decision on the irreducible polynomial, $x^3 \equiv x+1$, this is used in constructing the multiplication table.

Addition Table

$+$	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
0	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
1	1	0	$x+1$	x	x^2+1	x^2	x^2+x+1	x^2+x
x	x	$x+1$	0	1	x^2+x	x^2+x+1	x^2	x^2+1
$x+1$	$x+1$	x	1	0	x^2+x+1	x^2+x	x^2+1	x^2
x^2	x^2	x^2+1	x^2+x	x^2+x+1	0	1	x	$x+1$
x^2+1	x^2+1	x^2	x^2+x+1	x^2+x	1	0	$x+1$	x
x^2+x	x^2+x	x^2+x+1	x^2	x^2+1	x	$x+1$	0	1
x^2+x+1	x^2+x+1	x^2+x	x^2+1	x^2	$x+1$	x	1	0

Multiplication Table

$\cdot$	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1	
0	0	0	0	0	0	0	0	0	0
1	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1	
x	0	x	x^2	x^2+x	$x+1$	1	x^2+x+1	x^2+1	
$x+1$	0	$x+1$	x^2+x	x^2+1	x^2+x+1	x^2	1	x	
x^2	0	x^2	$x+1$	x^2+x+1	x^2+x	x	x^2+1	1	
x^2+1	0	x^2+1	1	x^2	x	x^2+x+1	$x+1$	x^2+x	
x^2+x	0	x^2+x	x^2+x+1	1	x^2+1	$x+1$	x	x^2	
x^2+x+1	0	x^2+x+1	x^2+1	x	1	x^2+x	x^2	$x+1$	

3 RSA encryption

RSA is one of the oldest versions of public key cryptosystems. Cryptosystems are widely used to have a secure way for (digital) communication. Public key cryptography, also known as asymmetric cryptography, is characterized by the use of two different keys; a public key and a private key. Two uses of public key cryptography are public key encryption for message exchange and digital signatures. The first version of the RSA encryption algorithm has been published by Ron Rivest, Adi Shamir and Leonard Adleman in 1978[10], the encryption method they suggested was used for digital signatures. RSA encryption can also be used to encrypt and decrypt messages so anyone eavesdropping the conversation can not uncover the original message. Until this day, no method has been published that would break this systems security as long as the key used is large enough. Breaking the security of the RSA encryption is known as the RSA problem.

3.1. Digital signatures

We will start with explaining how the RSA algorithm works according to Rivest, Shamir and Adleman. They created an algorithm to add digital signatures, this algorithm consists of two parts; the encryption algorithm E and the decryption algorithm D.

When they developed this method they agreed on four necessary properties any public key cryptosystem has to satisfy;

- Deciphering the enciphered form of a message M returns the original message M. Formally,

$$D(E(M)) = M. \tag{3.1}$$

- Both E and D are easy to compute.
- By publicly revealing E, no easy way to compute D will be revealed. This means in practice that only the owner of the private key can decrypt messages encrypted with E, or compute D efficiently.
- If a message M is first deciphered and then enciphered, the message M is the result. Formally,

$$E(D(M)) = M. \tag{3.2}$$

Now we know what properties the cryptosystem has to satisfy, let's discuss how RSA encryption works.

As we often do in the field of cryptography, we consider two persons called Alice (A) and Bob (B) who will use the system. Both Alice and Bob have an encryption and a decryption procedure, which we call E_A, D_A, E_B and D_B (the encryption and decryption procedure for Alice and the encryption and decryption procedure for Bob respectively). For the encryption algorithms, both Alice and Bob have their own public key. For the decryption they both have a private key. Assume Bob wants to send Alice a signed message. He starts by computing the signature for the message M;

$$S = D_B(M)$$

Instead of sending S to Alice, Bob computes $C = E_A(S)$ and sends the result to Alice. If Bob would send S instead, everyone could apply Bob's (public) encryption algorithm to read the original message M. By sending C, only Alice can decrypt the original message. After receiving the ciphertext C, Alice decrypts the ciphertext with her decryption algorithm to obtain S;

$$S = D_A(E_A(S)).$$

To read the original message, all Alice needs to do is compute $E_B(S)$, which she can do since the key for Bob's encryption algorithm is a public key;

$$M = E_B(D_B(M)) = E_B(S).$$

The message exchange procedure described above is visualized in figure 3.1.

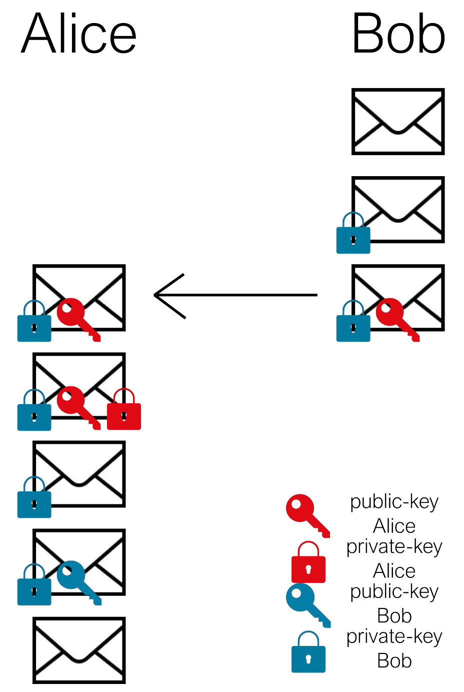


Figure 3.1: Visualization of the RSA encryption algorithm

Following these steps ensures Alice she has received a message from Bob that has not been altered by any eavesdropper; Bob is the (single) owner of the private key and therefore he must have been the one to encrypt the message. Moreover, for this same reason Bob cannot deny he was the sender of the message that Alice received.

3.2. Message exchange

A similar scheme as used for digital signatures can be applied to exchange messages. Assume Alice (A) wants to send a message M to Bob (B), but she does not want anyone eavesdropping on the conversation to be able to decrypt the message she sent. Alice starts by computing $C = E_B(M)$ and sends C to Bob. The key used for encryption is the public key, so anyone could compute C. However, the key for decryption is a private key (in this situation Bob's private key). Only individuals with the private key can compute M after receiving the ciphertext C

$$M = D_B(C) = D_B(E_B(M)).$$

Figure 3.2 shows a visualisation of this process.

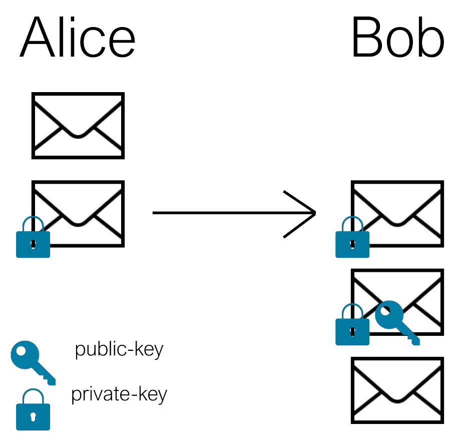


Figure 3.2: Visualisation of message exchange

3.3. Mathematics

We will once again describe message exchange using RSA encryption. This time using mathematics. This process is represented in figure 3.3. Here Bob sends a message to Alice. We will devide the process in three steps; key generation, encryption and decryption.

Key generation is performed by Alice; Alice starts with randomly selecting two large enough primes p and q and uses these to calculate $n = p \cdot q$. The following step will be selecting some positive integer $e < n$ such that $gcd(e, \phi(n)) = 1$ and calculating $d = e^{-1}(mod\ \phi(n))$. e is the public key and d will be refered to as the private key.

Encryption is done by Bob; he starts with receiving the public key (e, n) from Alice. Before he can encrypt the message he wants to send to Alice he needs to represent the message as some integer M with $1 \leq M < n$. Bob now has everything to compute the ciphertext $C \equiv M^e\ (mod\ n)$, which he sends back to Alice.

Decryption is again executed by Alice. She uses the private key to compute $M \equiv C^d\ (mod\ n)$, and extract the original plaintext from the integer M.

$$\begin{array}{lcr}
\text{Alice} & & \text{Bob} \\
p, q \overset{R}{\leftarrow} \mathbb{Z}_p & & \\
n = p \cdot q & & \\
e \overset{R}{\leftarrow} \mathbb{Z}_n; \gcd\big(e, \phi(n)\big) = 1 & & \\
d;\ e \cdot d \equiv 1 \pmod{\phi(n)} & & \\
& \xrightarrow{(e,n)} & \\
& & M \overset{R}{\leftarrow} \mathbb{Z}_n \\
& & C \equiv M^e \pmod{n} \\
& \xleftarrow{C} & \\
M \equiv C^d \pmod{n} & &
\end{array}$$

Figure 3.3: Message exchange with RSA encryption

Now it is time to prove the validity of this encryption. To use the method described above, one starts with selecting two very large (random) primes p and q and calculates $n = p \cdot q$. To encrypt the message

M, it needs to be represented as an integer in $\{0, 1, ..., n-1\}$ such that M and n are coprime. Our public key is a pair (e, n) and our private key is another pair (d, n). Equation (3.3) describes the encryption and decryption formula used.

$$\begin{aligned} E(M) &\equiv M^e \text{ (mod } n) \\ D(M) &\equiv M^d \text{ (mod } n) \end{aligned} \tag{3.3}$$

In order to make sure the recipient encodes the correct message, it is important that Equation (3.1) and Equation (3.2) both hold.

Proposition 3.3.1. *$D(E(M)) = M^{e \cdot d} \equiv M \ (mod\ n)$ and $E(D(M)) = M^{d \cdot e} \equiv M \ (mod\ n)$.*

Proof. Since multiplication is commutative we can write $D(E(M) = E(D(M)) \equiv M^{e*d} \ (mod\ n)$, one needs

$$M^{e \cdot d} \equiv M \ (mod\ n).$$

Assume M and n are coprime, Euler's Theorem, Theorem 2.1.18, implies $M^{\phi(n)} \equiv 1 \ (mod\ n)$. By Euler's product formula, Theorem 2.1.17, $\phi(n) = \phi(p) \cdot \phi(q) = n - (p+q) + 1 = (p-1)(q-1)$.
d is chosen such that $gcd(d, (p-1)(q-1)) = 1$. e needs to be such that Equation (3.4) holds

$$e \cdot d \equiv 1 \ (mod\ (p-1)(q-1)). \tag{3.4}$$

Since $gcd(d, (p-1)(q-1)) = 1$, we have $d \in (\mathbb{Z}/\phi(n)\mathbb{Z})^*$. By the definition of groups, there must exists an e such that Equation (3.4) holds.
Since $e \cdot d \equiv 1 \ (mod\ \phi(n))$, we can write $e \cdot d = k \cdot \phi(n) + 1$ for some $k \in \mathbb{Z}$. Together with Fermat's Little Theorem, Theorem 2.2.1, Equation (3.5) can be established.

$$\begin{aligned} M^{p-1} &\equiv 1 \text{ (mod } p),\ (p-1) \mid \phi(n) \implies M^{k\phi(n)+1} \equiv M \text{ (mod } p) \\ M^{q-1} &\equiv 1 \text{ (mod } q),\ (q-1) \mid \phi(n) \implies M^{k\phi(n)+1} \equiv M \text{ (mod } q) \end{aligned} \tag{3.5}$$

As a result of Equation (3.4) and Equation (3.5), $M^{e \cdot d} \equiv M^{k\phi(n)+1} \equiv M \ (mod\ n)$.

Notice that Equation (3.1) and Equation (3.2) still hold when M and n are not coprime. Assume M and n are not coprime, that is $gcd(M, n) \neq 1$. Since $n = p \cdot q$ with p and q both prime, this would mean that either $p \mid M$ or $q \mid M$ (not both). WLOG assume $p \mid M$. It is trivial that now $M^{e \cdot d} = M^{k\phi(n)+1} \equiv M \ (mod\ p)$. The same method as before can be used to show $M^{e \cdot d} = M^{k\phi(n)+1} \equiv M \ (mod\ q)$. Hence $M^{e \cdot d} \equiv M \ (mod\ n)$. □

Even though theoretically our public key e could be a small number, e is often large. For the calculation of $C \equiv M^e$, an algorithm like the left-to-right binary method is often used to speed up the process of calculating the (high) power. Algorithm 3.3.2 explains how this algorithm works. Common choices for e are $3, 5, 17, 257$ and 65537 since these can all be written as $2^k + 1$ for some $k \in \mathbb{N}$ and are prime. Example 3.3.3 gives an example of how Algorithm 3.3.2 works.

Algorithm 3.3.2 (left-to-right binary method)**.** *This algorithm is one of the ways to reduce the number of calculations necessary to compute $r = k^n$ for $k \in \mathbb{Z}$ and $n \in \mathbb{N}$.*

- *Write n in binary.*
- *Initialize $r \leftarrow 1$.*
- *Go through the bits representing n in binary from left to right.*
  If $bit = 0$, set $r \leftarrow r^2$ and continue with the next bit.
  If $bit = 1$, start with $r \leftarrow r^2$. Then set $r \leftarrow r \cdot k$ and continue with the next bit.

Example 3.3.3 (left-to-right binary method)**.** *To compute 23^{17} using the left-to-right binary method, we start with writing 17 as a binary number; $17 = 10001$. Initialize $r \leftarrow 1$.*

- *Bit $1 = 1$, $r \leftarrow r^2 = 1$, then set $r \leftarrow r \cdot k = 23$.*
- *Bit $2 = 0$, $r \leftarrow r^2 = 529$.*
- *Bit $3 = 0$, $r \leftarrow r^2 = 279\ 841$.*

- *Bit* $4 = 0, r \leftarrow r^2 = 7\ 831\ 098\ 281.$
- *Bit* $5 = 1, r \leftarrow r^2 = 6\ 132\ 610\ 415\ 680\ 998\ 648\ 961,$ *then set* $r \leftarrow r \cdot k = 141\ 050\ 039\ 560\ 662\ 968\ 926\ 103.$

Using the left-to-right binary method we have computed $23^{17} = 141\ 050\ 039\ 560\ 662\ 968\ 926\ 103$ *using* 7 *computations.*

Now we know how to compute the encryption and the decryption is it necessary to know how to chose the public key e and the private key d. Let d be an integer such that $gcd(d, (p-1)(q-1)) = 1$. To determine the public key to execute RSA encryption we need to find e such that $e \cdot d \equiv 1\ (mod\ (p-1)(q-1))$. This means that e and d are each others inverses modulo $\phi(n)$, Corollary 2.1.2 and a modification of Algorithm 2.1.3 will be used to find e. By Corollary 2.1.2, which is a special case of Bezout's Identity, there must exist integers x and y such that $xd + y\phi(n) = 1$. After finding these x and y we can take $e = x$. To find these x and y, start by performing the Euclidean Algorithm, Algorithm 2.1.3, with $a = \phi(n)$ and $b = d$. When using this algorithm to find the greatest common divider of two integers it is not necessary to remember the values for the q_i's. However, this time it is necessary to safe those integers alongside the sequence $r_0, r_1, \ldots$. We can stop the algorithm as soon as we find $r_i = 1$ for some $i \in \mathbb{N}$. We use substitution to express this r_i using q_j with $0 < j < i$ and r_0 and r_1. Example 3.3.4 shows how one can find the inverse of an element in the group $(\mathbb{Z}/n\mathbb{Z})^*$.

Example 3.3.4 (Inverse of 100 modulo 257). $gcd(257, 100) = 1$, *therefore* $100 \in (\mathbb{Z}/257\mathbb{Z})^*$ *and there must exist some* $100^{-1} \in (\mathbb{Z}/257\mathbb{Z})^*$ *such that* $100 \cdot 100^{-1} \equiv 1\ (mod\ 257)$.

- *Set* $r_0 = 257$ *and* $r_1 = 100$.
- $r_2 = r_0 - q_1 r_1$, *thus* $57 = 257 - 2 \cdot 100$.
- $r_3 = r_1 - q_2 r_2$, *thus* $43 = 100 - 1 \cdot 57$.
- $r_4 = r_2 - q_3 r_3$, *thus* $14 = 57 - 1 \cdot 43$.
- $r_5 = r_3 - q_4 r_4$, *thus* $1 = 43 - 3 \cdot 14$.

Now use substitution to find x *and* y *such that* $100x + 257y = 1$

$$\begin{aligned} 1 &= 43 - 3 \cdot 14 \\ &= 43 - 3(57 - 1 \cdot 43) = -3 \cdot 57 + 4 \cdot 43 \\ &= -3 \cdot 57 + 4(100 - 1 \cdot 57) = 4 \cdot 100 - 7 \cdot 57 \\ &= 4 \cdot 100 - 7(247 - 2 \cdot 100) = -7 \cdot 257 + 18 \cdot 100. \end{aligned}$$

We have found $y = -7$ *and* $x = 18 = 100^{-1}$ *(one can easily confirm that* 18 *is indeed the inverse of* 100 *modulo* 257*).*

4 Semifields

This chapter will be on the implementation of RSA encryption on a number of mathematical structures. The focus will be on checking whether encryption E and decryption D are well defined as in Equation (4.1) and Equation (4.2).

$$E(M) \equiv M^e \tag{4.1}$$

$$D(M) \equiv M^d \tag{4.2}$$

One example of a category of so-called exotic fields are the semifields constructed by Knuth[7].

Definition 4.0.1 (Semifield)**.** *A finite semifield S is a finite algebraic system containing at least two elements. S possesses two binary operations, addition and multiplication, which satisfy the following axioms*

A1 *Addition is a group, with identity element 0.*

A2 *If $ab = 0$, then either $a = 0$ or $b = 0$.*

A3 $a(b + c) = ab + ac$*;* $(a + b)c = ac + bc$*.*

A4 *There is an element 1 in S such that $1a = a1 = a$.*

Within the category of semifields, there is the category of proper semifields;

Definition 4.0.2 (Proper semifield)**.** *A proper semifield is a semifield which is not a field.*

Definition 4.0.3 (Pre-semifield)**.** *A system is a pre-semifield if all conditions for semifields are satisfied except possibly $A4$.*

Example 4.0.4 (Proper semifield)**.** *A proper semifield V will be constructed and the possibility to use RSA encryption on the field will be analyzed.*

$$V = \{u + \lambda v;\ u,\ v \in GF(4)\}$$

The operations addition and multiplication are both defined on V as follows:

$$(u + \lambda v) + (x + \lambda y) = (u + x) + \lambda(v + y)$$

$$(u + \lambda v)(x + \lambda y) = (ux + v^2 y) + \lambda(vx + u^2 y + v^2 y^2).$$

Here, the addition and multiplication of u, v, x and y is defined as multiplication and addition in $GF(4)$. V is indeed a proper semifield. All the properties $A1$,$A2$, $A3$ and $A4$ are satisfied;

- *Property $A1$ holds. Since u, v, x and y are elements of $GF(4)$, both $u + x$ and $v + y$ are in $GF(4)$ and therefore $(u + \lambda v) + (x + \lambda y) = (u + x) + \lambda(v + y) \in V$.*

- *To get $(u + \lambda v)(x + \lambda y) = 0 + 0$, we require both $ux + v^2y = 0$ and $vx + u^2y + v^2y^2 = 0$. If $ux + v^2y = 0$ and none of the original factors is zero, then there must be some z (unequal to zero) such that $x = v^2z$ and $y = uz$. Then, $vx + u^2y + v^2y^2 = v^3z + u^3z + v^2u^2z^2 = 0$. This would imply $u = v = 0$, which gives a contradiction. Therefore $A2$ must hold.*
- *Write $a = u + \lambda v$, $b = x + \lambda y$ and $c = p + \lambda q$. One can show that $a(b + c) = ab + ac = (ux + up + v^2y + v^2q) + \lambda(vx + vp + u^2y + u^2q + v^2y^2 + v^2q^2)$. Something similar holds for $(a + b)c = ac + bc$. This implies that property $A3$ is valid for V.*
- *Take $1 = 1 + \lambda 0 \in V$. Then $1a = a1 = a$ for any $a \in V$. Hence property $A4$ is satisfied.*

Moreover, if $a = t+\lambda 0$, $b = t+\lambda 1$ and $c = t+\lambda t$, then $(ab)c = t+\lambda t^2$ and $a(bc) = t+\lambda 0$. Therefore the system V is not only a semifield but a proper semifield. Multiplication as defined for system V is not associative as can be found in 4.1. This multiplication table is optained using Python (see Appendix A).

	$(0,0)$	$(0,1)$	$(0,t)$	$(0,t^2)$	$(1,0)$	$(1,1)$	$(1,t)$	$(1,t^2)$	$(t,0)$	$(t,1)$	(t,t)	(t,t^2)	$(t^2,0)$	$(t^2,1)$	(t^2,t)	(t^2,t^2)
$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$
$(0,1)$	$(0,0)$	$(1,1)$	(t,t^2)	(t^2,t)	$(0,1)$	$(1,0)$	(t,t)	(t^2,t^2)	$(0,t)$	$(1,t^2)$	$(t,1)$	$(t^2,0)$	$(0,t^2)$	$(1,t)$	$(t,0)$	$(t^2,1)$
$(0,t)$	$(0,0)$	(t^2,t^2)	$(1,t)$	$(t,1)$	$(0,t)$	$(t^2,1)$	$(1,0)$	(t,t^2)	$(0,t^2)$	$(t^2,0)$	$(1,1)$	(t,t)	$(0,1)$	(t^2,t)	$(1,t^2)$	$(t,0)$
$(0,t^2)$	$(0,0)$	(t,t)	$(t^2,1)$	$(1,t^2)$	$(0,t^2)$	$(t,1)$	(t^2,t)	$(1,0)$	$(0,1)$	(t,t^2)	$(t^2,0)$	$(1,t)$	$(0,t)$	$(t,0)$	(t^2,t^2)	$(1,1)$
$(1,0)$	$(0,0)$	$(0,1)$	$(0,t)$	$(0,t^2)$	$(1,0)$	$(1,1)$	$(1,t)$	$(1,t^2)$	$(t,0)$	$(t,1)$	(t,t)	(t,t^2)	$(t^2,0)$	$(t^2,1)$	(t^2,t)	(t^2,t^2)
$(1,1)$	$(0,0)$	$(1,0)$	$(t,1)$	$(t^2,1)$	$(1,1)$	$(0,1)$	$(t^2,0)$	$(t,0)$	(t,t)	(t^2,t)	$(0,t^2)$	$(1,t^2)$	(t^2,t^2)	(t,t^2)	$(1,t)$	$(0,t)$
$(1,t)$	$(0,0)$	(t^2,t)	$(1,0)$	(t,t)	$(1,t)$	$(t,0)$	$(0,t)$	$(t^2,0)$	(t,t^2)	$(1,1)$	(t^2,t^2)	$(0,1)$	$(t^2,1)$	$(0,t^2)$	$(t,1)$	$(1,t^2)$
$(1,t^2)$	$(0,0)$	(t,t^2)	(t^2,t^2)	$(1,0)$	$(1,t^2)$	$(t^2,0)$	$(t,0)$	$(0,t^2)$	$(t,1)$	$(0,t)$	$(1,t)$	$(t^2,1)$	(t^2,t)	$(1,1)$	$(0,1)$	(t,t)
$(t,0)$	$(0,0)$	$(0,t^2)$	$(0,1)$	$(0,t)$	$(t,0)$	(t,t^2)	$(t,1)$	(t,t)	$(t^2,0)$	(t^2,t^2)	$(t^2,1)$	(t^2,t)	$(1,0)$	$(1,t^2)$	$(1,1)$	$(1,t)$
$(t,1)$	$(0,0)$	$(1,t)$	(t,t)	$(t^2,0)$	$(t,1)$	(t^2,t^2)	$(0,t^2)$	$(1,1)$	(t^2,t)	$(t,0)$	$(1,0)$	$(0,t)$	$(1,t^2)$	$(0,1)$	$(t^2,1)$	(t,t^2)
(t,t)	$(0,0)$	$(t^2,0)$	$(1,t^2)$	(t,t^2)	(t,t)	$(1,t)$	$(t^2,1)$	$(0,1)$	(t^2,t^2)	$(0,t^2)$	$(t,0)$	$(1,0)$	$(1,1)$	$(t,1)$	$(0,t)$	(t^2,t)
(t,t^2)	$(0,0)$	$(t,1)$	$(t^2,0)$	$(1,1)$	(t,t^2)	$(0,t)$	$(1,t^2)$	(t^2,t)	$(t^2,1)$	$(1,0)$	$(0,1)$	$(t,0)$	$(1,t)$	(t^2,t^2)	(t,t)	$(0,t^2)$
$(t^2,0)$	$(0,0)$	$(0,t)$	$(0,t^2)$	$(0,1)$	$(t^2,0)$	(t^2,t)	(t^2,t^2)	$(t^2,1)$	$(1,0)$	$(1,t)$	$(1,t^2)$	$(1,1)$	$(t,0)$	(t,t)	(t,t^2)	$(t,1)$
$(t^2,1)$	$(0,0)$	$(1,t^2)$	$(t,0)$	(t^2,t^2)	$(t^2,1)$	(t,t)	$(1,1)$	$(0,t)$	$(1,t)$	$(0,1)$	(t^2,t)	$(t,1)$	(t,t^2)	$(t^2,0)$	$(0,t^2)$	$(1,0)$
(t^2,t)	$(0,0)$	$(t^2,1)$	$(1,1)$	$(t,0)$	(t^2,t)	$(0,t^2)$	(t,t^2)	$(1,t)$	$(1,t^2)$	(t,t)	$(0,t)$	(t^2,t^2)	$(t,1)$	$(1,0)$	$(t^2,0)$	$(0,1)$
(t^2,t^2)	$(0,0)$	$(t,0)$	(t^2,t)	$(1,t)$	(t^2,t^2)	$(1,t^2)$	$(0,1)$	$(t,1)$	$(1,1)$	$(t^2,1)$	(t,t^2)	$(0,t^2)$	(t,t)	$(0,t)$	$(1,0)$	$(t^2,0)$

Figure 4.1: Multiplication table system V. $(u,v) = u + \lambda v$, cell (X,Y) has value $Y \cdot X$.

Despite multiplication not being associative, powers still might be defined (i.e. the left $(X(XX))$ and right $((XX)X)$ powers are equal). A quick analysis of the table of both left and right powers (see Appendix B) obtained using Python code (see Appendix A) tells us that for this system V the left and right powers are not necessarily equal and therefore powers are not well defined. An example of an element in V for which the left and right power are different is the element $x = t+\lambda 1$, as for this element $x(xx) = t^2 + \lambda t$ while $(xx)x = t^2 + \lambda t^2$. If we would have $x(xx) = (xx)x$ for all $x \in V$, then the powers are well defined and we would say that the structure V is power associative.

In Example 4.0.4 we have seen that whenever we are dealing with a proper semifield, this field needs to be power associative in order to extend the RSA algorithm onto this field. We will consider a structure that is power associative.

Example 4.0.5 (Power associative)**.** *Let S be the set $S = \{0, u, v, u + v\}$. The operations addition and multiplication are defined as in the addition and multiplication tables.*

Addition Table

$+$	0	u	v	$u+v$
0	0	u	v	$u+v$
u	u	0	$u+v$	v
v	v	$u+v$	0	u
$u+v$	$u+v$	v	u	0

Multiplication Table

$\cdot$	0	u	v	$u+v$
0	0	0	0	0
u	0	u	$u+v$	v
v	0	$u+v$	v	u
$u+v$	0	v	u	$u+v$

To see that the system S is not muliplicative associative, take the elements u, v and $u+v$. Then $u\cdot(v\cdot(u+v)) = u$ while $(u\cdot v)\cdot(u+v) = u+v$. However, it is clear that S is power associative: for all $x \in S$, $x(xx) = x = (xx)x$. While implementing RSA encryption on the system S as in Equation (4.1) and Equation (4.2), the values for the public key e and secret key d can be chosen arbitrarily from the set of natural numbers since $x^n = x$ for all $x \in S$ and all $n \in \mathbb{N}$.

The system S as defined in Example 4.0.5 does not have a multiplicative identity and therefore does not satisfy condition $A4$, hence S is not a (proper) semifield. However, it is rather easy to show that the other three properties for semifields are satisfied by the system S. As a result this system is a pre-semifield.

Another pre-semifield given by Knuth[7] is the one derived from any field F with more than one automorphism (thus an automorphism that is not the identity). A possible choice for an automorphism is the Frobenius Automorphism.

Definition 4.0.6 (Frobenius Automorphism). *Let F be the finite field with characteristic p and $q = p^n$ elements, thus $F = GF(q)$. Take $K = GF(q^N)$ for some fixed $N \in \mathbb{N}$. For all $a \in K$, the Frobenius Automorphism of K over F is defined as*

$$\sigma(a) = a^q$$

The first step in proving wheter the Frobenius Automorphism is indeed an automorphism on any finite field is to proof that the Frobenius Automorphism preserves the structure of a finite field, i.e. it preserves addition and multiplication.

Proposition 4.0.7. *The Frobenius Automorphism perserves addition and multiplication, that is for any $a, b \in F$,*

$$\sigma(a+b) = \sigma(a) + \sigma b$$
$$\sigma(a\cdot b) = \sigma(a)\cdot\sigma(b)$$

Proof. We start with proving that the Frobenius Automorphism preserves addition. Let $GF(q)$ be a finite field with $q = p^n$ for some prime p and some $n \in \mathbb{N}$. Take $a, b \in GF(q)$ arbitrary. Using the Binomium of Newton to write

$$\begin{aligned}\sigma(a+b) &= (a+b)^q\\ &= \sum_{k=0}^{q}\binom{q}{k}a^{q-k}b^k\\ &= a^q + \binom{q}{1}a^{q-1}b + \dots + \binom{q}{q-1}ab^{q-1} + b^q.\end{aligned}$$

If $k = 1, 2, \dots, q-1$, then $\binom{q}{k} = \frac{q\cdot(q-1)!}{k!(q-k)!}$. Hence for $k = 1, 2, \dots, q-1$, $\binom{q}{k}a^{q-k}b^k = \frac{q\cdot(q-1)!}{k!(q-k)!}a^{q-k}b^k \equiv 0$ in the field $GF(q)$. Therefore $\sigma(a+b) = (a+b)^q = a^q + \binom{q}{1}a^{q-1}b + \dots + \binom{q}{q-1}ab^{q-1} + b^q \equiv a^q + b^q$ in the finite field, which is exactly equal to $\sigma(a) + \sigma(b)$.

For multiplication we have $\sigma(a\cdot b) = (a\cdot b)^q = a^q\cdot b^q = \sigma(a)\cdot\sigma(b)$. □

Next, we will prove that the Frobenius Automorphism is a bijection.

Proposition 4.0.8. *The Frobenius Automorphism is a bijective function.*

Proof. Let e be the identity element of an arbitrary finite field F with corresponding Frobenius Automorphism σ. Since the Frobenius Automorphism preserves multiplication, we have $\sigma(ee) = \sigma(e) = \sigma(e)\sigma(e)$, which only holds when $e = \sigma(e)$. Then by Theorem 2.1.11, the Frobenius Automorphism is injective. Since the domain and the codomain of the Frobenius Automorphism both are the finite field F and therefore have the same number of elements, the Frobenius Automorphism must be bijective. □

Now we can conclude that the Frobenius Automorphism is indeed an automorphism on a finite field. The next example will show how the Frobenius Automorphism works on the finite fields $GF(9)$ and $GF(8)$ over $GF(3)$ and $GF(@)$ respectively.

Example 4.0.9 ($GF(9)$). *$GF(9)$ is the finite field with prime characteristic $char(GF(9)) = 3$. By definition the Frobenius Automorphism $\sigma : GF(9) \to GF(9)$ is*

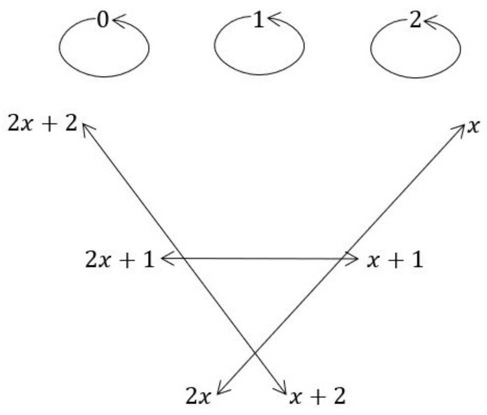
$$\sigma(a) = a^3$$

for all $a \in GF(9)$.

Figure 4.2 shows how the automorphism acts on the elements of the field.

a	$\sigma(a)$
0	0
1	1
2	2
x	$2x$
$x+1$	$2x+1$
$x+2$	$2x+2$
$2x$	x
$2x+1$	$x+1$
$2x+2$	$x+2$

(a)



(b)

Figure 4.2: σ on $GF(9)$

Notice that for 0, 1 and 2, $\sigma(a) = a$. This can be generalized as in the next theorem.

Theorem 4.0.10. *Let $F = GF(q)$ with $q = p^n$ be a finite field with $char(F) = p$ and let σ be the corresponding Frobenius Algorithm. Then $\sigma(a) = a$ if and only if $a \in \mathbb{Z_p} \cong \{0, 1, \dots, p-1\}$.*

Proof. By Theorem 2.3.4, $ord(F) = q$. Therefore F has a subfield K that is isomorpic to $\{0, 1, \dots, p-1\}$. By Fermat's Little Theorem, Theorem 2.2.1, for all $a \in K$ we have $\sigma(a) = a^p \equiv a$ within the field F. □

Example 4.0.11 ($GF(8)$). *$GF(8)$ is the finite field with prime characteristic $char(GF(8)) = 2$. Since $8 = 2^3$, there are two options σ_1 and σ_2 for the Frobenius Automorphism over $GF(2)$ and $GF(4)$ respectively. By definition the Frobenius Automorphisms $\sigma_1, \sigma_2 : GF(8) \to GF(8)$ are*

$$\sigma_1(a) = a^2$$
$$\sigma_2(a) = a^4$$

for all $a \in GF(8)$.

Figure 4.3 and Figure 4.4 show how the automorphisms acts on the elements of the field.

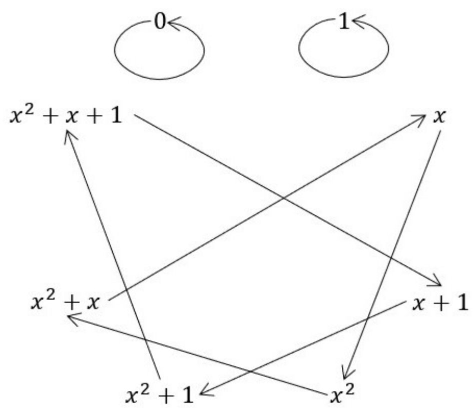
Using the Frobenius Automorphism we are able to construct the structure $(F, +, \circ)$ where F is a finite field and the operation $\circ$ is defined as $x \circ y = \sigma(xy)$ for all $x, y \in F$. Notice that the multiplication of x and y on the right-hand side takes place in the field F. We will show that the structure $(F, +, \circ)$ is a pre-semifield;

A1 Since F is a field and addition takes place in this field, addition is a group with identity element 0.

A2 Assume $a \circ b = 0$. Then by Theorem 4.0.10 and Proposition 4.0.7, $a \circ b = \sigma(ab) = \sigma(a)\sigma(b) = 0$. Because the multiplication of $\sigma(a)$ and $\sigma(b)$ takes place in the field F and not in the structure $(F, +, \circ)$, we must have either $\sigma(a) = 0$ or $\sigma(b) = 0$. This is equivalent to saying either $a \in \ker(\sigma)$ or $b \in \ker(\sigma)$. Since $ker(\sigma) = 0$, either $a = 0$ or $b = 0$.

a	$\sigma_1(a)$
0	0
1	1
x	x^2
$x+1$	x^2+1
x^2	x^2+x
x^2+1	x^2+x+1
x^2+x	x
x^2+x+1	$x+1$

(a)

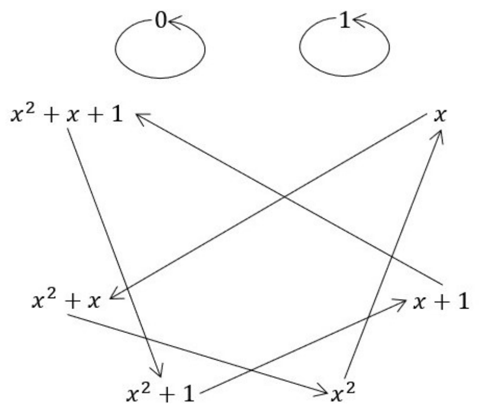


(b)

Figure 4.3: σ_1 on $GF(8)$

a	$\sigma_2(a)$
0	0
1	1
x	x^2+x
$x+1$	x^2+x+1
x^2	x
x^2+1	$x+1$
x^2+x	x^2
x^2+x+1	x^2+1

(a)



(b)

Figure 4.4: σ_2 on $GF(8)$

A3 Let $a, b, c \in F$ be arbitrary. Then $a \circ (b+c) = \sigma(a(b+c)) = \sigma(ab+ab) = \sigma(ab)+\sigma(ac) = a \circ b + a \circ c$ by Proposition 4.0.7.

In order to ensure that it is possible to extend RSA encryption to the pre-semifield $(F, +, \circ)$, Equation (4.1) and Equation (4.2) must hold. This requires our pre-semifield to be power associative.

Proposition 4.0.12. *The pre-semifield $(F, +, \circ)$ with $x \circ y = \sigma(xy)$ for all $x, y \in F$ is power associative.*

Proof. Let $a \in F$ be arbitrary. Then by Proposition 4.0.7

$$\begin{aligned} a \circ (a \circ a) &= \sigma(a \cdot \sigma(aa)) \\ &= \sigma(a)\sigma(\sigma(aa)) \\ &= \sigma(\sigma(aa))\sigma(a) \\ &= \sigma(\sigma(aa) \cdot a) \\ &= (a \circ a) \circ a. \end{aligned}$$

□

5 Conclusion

This paper aimed to construct mathematical structures RSA encryption could be extended onto. Both the encryption and decryption of the RSA algorithm are based on computing a power of either the message M or the encrypted ciphertext C. The mathematical structures we decided to study were different types of (finite) semifields. A finite semifield S is a finite algebraic system containing at least two elements and possesses the operations addition and multiplication. Besides semifields, also proper semifields and pre-semifields have been discussed. What seems crucial in extending the RSA algorithms for encryption and decryption on these structures is whether powers are associative, i.e. $x(xx) = (xx)x$. It is possible for a structure to be power associative but not associative with respect to multiplication. The set $S = \{0, u, v, u + v\}$ with addition and multiplicatoin as in the addition and multiplication table is an example of a mathematical structure that is not multiplicative associative but is power associative.

Addition Table

$+$	0	u	v	$u+v$
0	0	u	v	$u+v$
u	u	0	$u+v$	v
v	v	$u+v$	0	u
$u+v$	$u+v$	v	u	0

Multiplication Table

$\cdot$	0	u	v	$u+v$
0	0	0	0	0
u	0	u	$u+v$	v
v	0	$u+v$	v	u
$u+v$	0	v	u	$u+v$

Furthermore we have seen that the structure $(F, +, \circ)$ is a pre-semifield and is power associative. Here F is a finite field, the operation $\circ$ is defined as $x \circ y = \sigma(xy)$ for all $x, y \in F$ and σ denotes the Frobenius algorithm.

This paper did not discuss the safety of the extension of RSA on these mathematical structures. Research on this topic is necessary before being able to use RSA as a cryptosystem for our digital world.

Bibliography

[1] R.E. Bishop. “on fermat’s little theorem”. In: (2008).

[2] D. Boneh and V. Shoup. “A Graduate Course in Applied Cryptography”. In: (Jan. 2020). version 0.5. URL: https://crypto.stanford.edu/~dabo/cryptobook/BonehShoup_0_5.pdf.

[3] D.C. Gijswijt. “Algebra 1”. In: (2021). Course AM1060, academic year 2020/2021.

[4] Haiman. “Notes on finite fields”. In: *Math 113* (2015). URL: https://math.berkeley.edu/~mhaiman/math113-summer15/finite-fields.pdf.

[5] J. Hilmar and C. Smyth. “Euclid meets Bezout: Intersecting algebraic plane curves with the Euclidean algorithm”. In: *American Mathematical Monthly* 117.3 (Mar. 2010), pp. 250–260. eprint: 0907.0361.

[6] B. Kaliski and J. Staddon. “PKCS 1: RSA Cryptography Specifications, Version 2.0”. In: *RSA Laboratories* (Oct. 1998). URL: https://www.rfc-editor.org/rfc/rfc2437.

[7] D.E. Knuth. “Finite Semifields and Projective Planes”. In: *Journal of Algebra* 2.2 (June 1965), pp. 182–217. ISSN: 0021-8693. DOI: https://doi.org/10.1016/0021-8693(65)90018-9. URL: https://www.sciencedirect.com/science/article/pii/0021869365900189.

[8] S.M. Lane and G. Birkhoff. *Algebra*. 3rd ed. AMS Chelsea Publishing Series. Chelsea Publishing Company, 1999. ISBN: 9780821816462. URL: https://books.google.nl/books?id=L6FENd8GHIUC.

[9] “Proofs about Frobenius”. In: (). URL: https://www-users.cse.umn.edu/~garrett/coding/Overheads/23_proofs.pdf.

[10] R.L. Rivest, A. Shamir, and L. Adleman. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. In: *Association for Computing Machinery* 21.2 (Feb. 1978), pp. 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342. URL: https://doi.org/10.1145/359340.359342.

[11] P.W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. DOI: 10.1137/s0097539795293172. URL: https://doi.org/10.1137/s0097539795293172.

[12] E.T. Whittaker and G.N. Watson. *A Course of Modern Analysis*. 4th ed. Cambridge Mathematical Library. Cambridge University Press, 1996. DOI: 10.1017/CBO9780511608759.

[13] B. Yan et al. “Factoring integers with sublinear resources on a superconducting quantum processor”. In: (Dec. 2022).

A

Python code

```python
import plotly.graph_objects as go
from tabulate import tabulate
import pandas as pd



### define addition function for semifield ###
def addition(x,y):
    x=str(x)
    y=str(y)

    valid_set = {"0", "1", "t", "t^2"}

    if x not in valid_set or y not in valid_set:
        return "Invalid input"

    elif x==y:
        return "0"
    elif x=="0":
        return y
    elif y=="0":
        return x
    elif {x,y}=={"1","t"}:
        return "t^2"
    elif {x,y}=={"1","t^2"}:
        return "t"
    else:
        return "1"



### define product function for semifield ###
def product(x,y):
    x=str(x)
    y=str(y)

    valid_set = {"0", "1", "t", "t^2"}

    if x not in valid_set or y not in valid_set:
        return "Invalid input"

    elif x=="0" or y=="0":
        return "0"
    elif x=="1":
        return y
    elif y=="1":
        return x
    elif x=="t" and y=="t":
        return "t^2"
    elif y=="t^2" and x=="t^2":
        return "t"
    else:
        return "1"



### define multiply function for semifield ###
def multiply(u,v,x,y):
    u=str(u)
    v=str(v)
    x=str(x)
    y=str(y)

    valid_set = {"0", "1", "t", "t^2"}

    if u not in valid_set or v not in valid_set or x not in valid_set or y not in valid_set:
        return "Invalid input"

    return [addition(product(u,x),product(product(v,v),y)),addition(addition(product(v,x),product(product(u,u),y)),product(product(v,v),product(y,y)))]



### create the pairs (u,v) ###

elements = ["0", "1", "t", "t^2"]
pairs = []

for x in elements:
    for y in elements:
        pair = (x, y)
        pairs.append(pair)

#print(pairs)



### headers and showindex ###
labels=[]

for pair in pairs:
    label='('+pair[0]+','+pair[1]+')'
    labels.append(label)

#print(labels)



### print multiplication table ###
multiplication_table = []
columns = pairs
rows = pairs

```

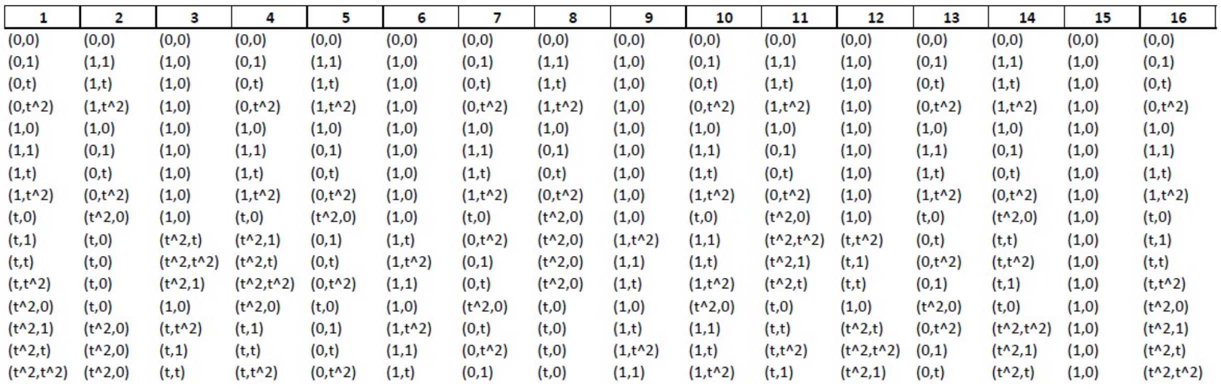
```
for r in rows:
    row = []
    for c in columns:
        value = multiply(r[0], r[1], c[0], c[1])
        entry = '('+','.join(value)+')'
        row.append(entry)
    multiplication_table.append(row)

print('table for multiplication')
print(tabulate(multiplication_table, headers=labels, showindex=labels))



### print powers table ###
powers_table = []
columns = list(range(1,17))
rows = pairs

for r in rows:
    row = ['('+r[0]+','+r[1]+')']
    value = r
    for i in range(1,16):
        value = multiply(value[0], value[1], r[0], r[1])
        entry = '('+','.join(value)+')'
        row.append(entry)
    powers_table.append(row)

print('table for powers')
print(tabulate(powers_table, headers=columns))



### print left powers table ###
left_powers_table = []
columns = list(range(1,17))
rows = pairs

for r in rows:
    row = ['('+r[0]+','+r[1]+')']
    value = r
    for i in range(1,16):
        value = multiply(r[0], r[1], value[0], value[1])
        entry = '('+','.join(value)+')'
        row.append(entry)
    left_powers_table.append(row)

print('table for left powers')
print(tabulate(left_powers_table, headers=columns))



### convert multiplication table to excel file ###
dataframe = pd.DataFrame(multiplication_table)
dataframe.index = pd.Index(labels)

file_path = r'C:\Users\lvanb\Documents\Bsc Applied Mathematics\BEP\code semifield\
    multiplication_table.xlsx'
dataframe.to_excel(file_path, header=labels)
```

```
### convert powers table to excel file ###
dataframe = pd.DataFrame(powers_table)

file_path = r'C:\Users\lvanb\Documents\Bsc Applied Mathematics\BEP\code semifield\
    powers_table.xlsx'
dataframe.to_excel(file_path, header = range(1,17))



### convert left powers table to excel file ###
dataframe = pd.DataFrame(left_powers_table)

file_path = r'C:\Users\lvanb\Documents\Bsc Applied Mathematics\BEP\code semifield\
    left_powers_table.xlsx'
dataframe.to_excel(file_path, header = range(1,17))
```

Listing A.1: Python code

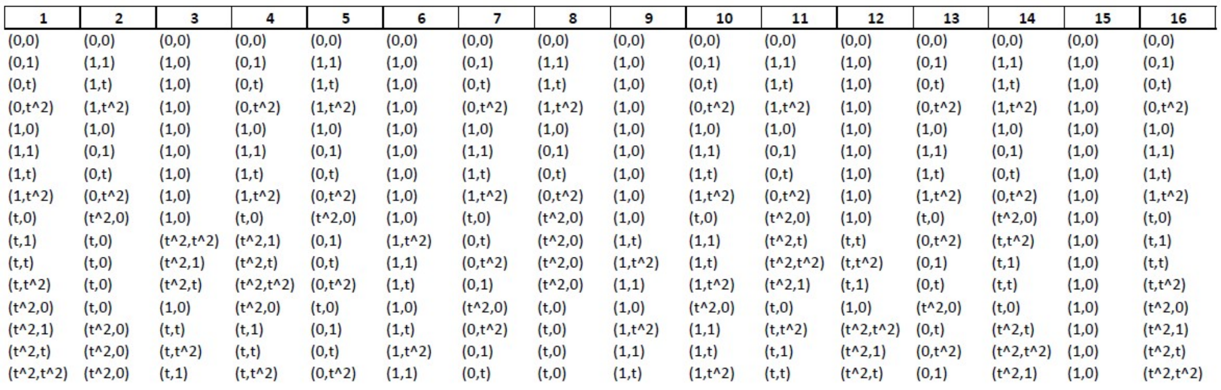
B Powers

Figure B.1 and Figure B.2 show the left ($x(xx)$) and the right ($(xx)x$) powers of the elements of structure V. Here $(u, v) = u + \lambda v$ and the header indicates to what power we raise the element.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)
(0,1)	(1,1)	(1,0)	(0,1)	(1,1)	(1,0)	(0,1)	(1,1)	(1,0)	(0,1)	(1,1)	(1,0)	(0,1)	(1,1)	(1,0)	(0,1)
(0,t)	(1,t)	(1,0)	(0,t)	(1,t)	(1,0)	(0,t)	(1,t)	(1,0)	(0,t)	(1,t)	(1,0)	(0,t)	(1,t)	(1,0)	(0,t)
(0,t^2)	(1,t^2)	(1,0)	(0,t^2)	(1,t^2)	(1,0)	(0,t^2)	(1,t^2)	(1,0)	(0,t^2)	(1,t^2)	(1,0)	(0,t^2)	(1,t^2)	(1,0)	(0,t^2)
(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)
(1,1)	(0,1)	(1,0)	(1,1)	(0,1)	(1,0)	(1,1)	(0,1)	(1,0)	(1,1)	(0,1)	(1,0)	(1,1)	(0,1)	(1,0)	(1,1)
(1,t)	(0,t)	(1,0)	(1,t)	(0,t)	(1,0)	(1,t)	(0,t)	(1,0)	(1,t)	(0,t)	(1,0)	(1,t)	(0,t)	(1,0)	(1,t)
(1,t^2)	(0,t^2)	(1,0)	(1,t^2)	(0,t^2)	(1,0)	(1,t^2)	(0,t^2)	(1,0)	(1,t^2)	(0,t^2)	(1,0)	(1,t^2)	(0,t^2)	(1,0)	(1,t^2)
(t,0)	(t^2,0)	(1,0)	(t,0)	(t^2,0)	(1,0)	(t,0)	(t^2,0)	(1,0)	(t,0)	(t^2,0)	(1,0)	(t,0)	(t^2,0)	(1,0)	(t,0)
(t,1)	(t,0)	(t^2,t)	(t^2,1)	(0,1)	(1,t)	(0,t^2)	(t^2,0)	(1,t^2)	(1,1)	(t^2,t^2)	(t,t^2)	(0,t)	(t,t)	(1,0)	(t,1)
(t,t)	(t,0)	(t^2,t^2)	(t^2,t)	(0,t)	(1,t^2)	(0,1)	(t^2,0)	(1,1)	(1,t)	(t^2,1)	(t,1)	(0,t^2)	(t,t^2)	(1,0)	(t,t)
(t,t^2)	(t,0)	(t^2,1)	(t^2,t^2)	(0,t^2)	(1,1)	(0,t)	(t^2,0)	(1,t)	(1,t^2)	(t^2,t)	(t,t)	(0,1)	(t,1)	(1,0)	(t,t^2)
(t^2,0)	(t,0)	(1,0)	(t^2,0)	(t,0)	(1,0)	(t^2,0)	(t,0)	(1,0)	(t^2,0)	(t,0)	(1,0)	(t^2,0)	(t,0)	(1,0)	(t^2,0)
(t^2,1)	(t^2,0)	(t,t^2)	(t,1)	(0,1)	(1,t^2)	(0,t)	(t,0)	(1,t)	(1,1)	(t,t)	(t^2,t)	(0,t^2)	(t^2,t^2)	(1,0)	(t^2,1)
(t^2,t)	(t^2,0)	(t,1)	(t,t)	(0,t)	(1,1)	(0,t^2)	(t,0)	(1,t^2)	(1,t)	(t,t^2)	(t^2,t^2)	(0,1)	(t^2,1)	(1,0)	(t^2,t)
(t^2,t^2)	(t^2,0)	(t,t)	(t,t^2)	(0,t^2)	(1,t)	(0,1)	(t,0)	(1,1)	(1,t^2)	(t,1)	(t^2,1)	(0,t)	(t^2,t)	(1,0)	(t^2,t^2)

Figure B.1: Left powers of structure V



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)
(0,1)	(1,1)	(1,0)	(0,1)	(1,1)	(1,0)	(0,1)	(1,1)	(1,0)	(0,1)	(1,1)	(1,0)	(0,1)	(1,1)	(1,0)	(0,1)
(0,t)	(1,t)	(1,0)	(0,t)	(1,t)	(1,0)	(0,t)	(1,t)	(1,0)	(0,t)	(1,t)	(1,0)	(0,t)	(1,t)	(1,0)	(0,t)
(0,t^2)	(1,t^2)	(1,0)	(0,t^2)	(1,t^2)	(1,0)	(0,t^2)	(1,t^2)	(1,0)	(0,t^2)	(1,t^2)	(1,0)	(0,t^2)	(1,t^2)	(1,0)	(0,t^2)
(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)	(1,0)
(1,1)	(0,1)	(1,0)	(1,1)	(0,1)	(1,0)	(1,1)	(0,1)	(1,0)	(1,1)	(0,1)	(1,0)	(1,1)	(0,1)	(1,0)	(1,1)
(1,t)	(0,t)	(1,0)	(1,t)	(0,t)	(1,0)	(1,t)	(0,t)	(1,0)	(1,t)	(0,t)	(1,0)	(1,t)	(0,t)	(1,0)	(1,t)
(1,t^2)	(0,t^2)	(1,0)	(1,t^2)	(0,t^2)	(1,0)	(1,t^2)	(0,t^2)	(1,0)	(1,t^2)	(0,t^2)	(1,0)	(1,t^2)	(0,t^2)	(1,0)	(1,t^2)
(t,0)	(t^2,0)	(1,0)	(t,0)	(t^2,0)	(1,0)	(t,0)	(t^2,0)	(1,0)	(t,0)	(t^2,0)	(1,0)	(t,0)	(t^2,0)	(1,0)	(t,0)
(t,1)	(t,0)	(t^2,t^2)	(t^2,1)	(0,1)	(1,t^2)	(0,t)	(t^2,0)	(1,t)	(1,1)	(t^2,t)	(t,t)	(0,t^2)	(t,t^2)	(1,0)	(t,1)
(t,t)	(t,0)	(t^2,1)	(t^2,t)	(0,t)	(1,1)	(0,t^2)	(t^2,0)	(1,t^2)	(1,t)	(t^2,t^2)	(t,t^2)	(0,1)	(t,1)	(1,0)	(t,t)
(t,t^2)	(t,0)	(t^2,t)	(t^2,t^2)	(0,t^2)	(1,t)	(0,1)	(t^2,0)	(1,1)	(1,t^2)	(t^2,1)	(t,1)	(0,t)	(t,t)	(1,0)	(t,t^2)
(t^2,0)	(t,0)	(1,0)	(t^2,0)	(t,0)	(1,0)	(t^2,0)	(t,0)	(1,0)	(t^2,0)	(t,0)	(1,0)	(t^2,0)	(t,0)	(1,0)	(t^2,0)
(t^2,1)	(t^2,0)	(t,t)	(t,1)	(0,1)	(1,t)	(0,t^2)	(t,0)	(1,t^2)	(1,1)	(t,t^2)	(t^2,t^2)	(0,t)	(t^2,t)	(1,0)	(t^2,1)
(t^2,t)	(t^2,0)	(t,t^2)	(t,t)	(0,t)	(1,t^2)	(0,1)	(t,0)	(1,1)	(1,t)	(t,1)	(t^2,1)	(0,t^2)	(t^2,t^2)	(1,0)	(t^2,t)
(t^2,t^2)	(t^2,0)	(t,1)	(t,t^2)	(0,t^2)	(1,1)	(0,t)	(t,0)	(1,t)	(1,t^2)	(t,t)	(t^2,t)	(0,1)	(t^2,1)	(1,0)	(t^2,t^2)

Figure B.2: Right powers of structure V