Universiteit Utrecht

Faculteit Bètawetenschappen

A cryptosystem based on algebraic surfaces

Bachelor Thesis

Emilin van Ruiten

Mathematics

Supervisor:

Dr. Valentijn Karemaker
Mathematical Institute

January 15, 2021

Abstract

This thesis discusses a cryptosystem based on algebraic surfaces, which was devised by Akiyama and Goto. At first the basics of cryptography, the P vs. NP problem and algebraic surfaces are discussed. Then the workings of the original version of the cryptosystem are explained, using a toy example to illustrate it. After that, an attack on the cryptosystem is explained, followed by a discussion of ways to resist the attack. The final part of this thesis explains the workings of an improved version of the cryptosystem, that is resistant to the attack. This explanation is also illustrated by a toy example.

Contents

Introduction

Throughout history the search for ways of secure communication has always been important. In the past, secure communication was mostly wanted for the purpose of warfare, so that commanders could send commands to their troops without the enemy being able to find out what those commands are. Nowadays, we have a lot of communication happening through the internet - where you never know who might be watching along with you - and thus secure communication is wanted for many more purposes. Banking, chatting and shopping are just a few examples of all the things we do online. Luckily, several different cryptosystems have already been invented for secure communication online. Most of the cryptosystems in use today are based on mathematical problems that are hard to solve, and therefore those cryptosystems are considered safe. However, with the rise of quantum computers the safety of those cryptosystems is in jeopardy, for quantum computation enables us to solve some of the hard mathematical problems on which the cryptosystems are based.

In this thesis we will look at a cryptosystem that is expected to be safe even when quantum computers have been fully developed. This cryptosystem, devised by Akiyama and Goto and described in [1], is based on the problem of finding sections on algebraic surfaces. Therefore we will call this cryptosystem the Algebraic Surface Cryptosystem, or ASC for short.

In the first chapter we will cover some preliminaries. We will begin this chapter by explaining the basics of cryptography and then we will briefly discuss the theory of what problems we consider hard to solve. After that we will explain the basics of algebraic surfaces and why the problem of finding sections on them is such a hard to solve problem. In the second chapter we explain the workings of the original version of the Algebraic Surface Cryptosystem, illustrated with a toy example. In the third chapter we explain an attack on the original version of the ASC, which makes it insecure. Then in the fourth and final chapter we will explain the workings of an improved version of the ASC that is resistant to the attack from chapter three. This explanation will once again be illustrated with a toy example.

Before we start, we would like to point out that we assume the reader has foreknowledge on fields and Galois theory. Furthermore, we would like to note that we will use the following notation throughout this thesis:

- $\mathbb{N} = \{0, 1, 2, ...\}$;
- $\mathbb{Z}^+ = \{1, 2, 3, ...\}$.

1 Preliminaries

1.1 Cryptography

The word cryptography is a composition of the Ancient Greek words κρυπτός, meaning 'hidden' or 'secret', and γράφειν, meaning 'to write', so the literal meaning of cryptography is something along the lines of 'hidden writing'. It may therefore not come as a suprise that cryptography is about finding ways to secure a message in such a way that its meaning stays hidden from any third party that might intercept the message. We will now illustrate this idea by means of an example.

Example 1.1.1. Suppose Alice wants to send a message to Bob. Of course she could just write it in normal English, put it in an envelope and send it to him, as she would with any message. However, this particular message contains confidential information that only Bob is allowed to know, and Bob shares his mailbox with his roommate Eve, who tends to be quite nosy and is therefore not unlikely to snoop around in Bob's mail. One way to prevent this from happening would be for Alice to personally hand the message to Bob.

Unfortunately though, there happens to be a pandemic going on, so it would wiser if Bob and Alice did not meet up. Therefore, Alice would like to make sure that even if Eve were to read the message, she would not be able to understand it. Luckily, the last time Alice and Bob did meet in person, they came up with their own secret alphabet, made up of symbols only they know the meaning of. Alice decides to write her original message in that secret alphabet, and then sends that version to Bob. Once Bob sees the message, he will recognize the alphabet and be able to retrieve the original message. Eve, on the other hand, will not be able to understand the secret alphabet, and will therefore not be able to discover the original message. Thus, Alice has succeeded. △

Besides the general notion of cryptography, Example 1.1.1 also illustrates some other important concepts of cryptography. We will give the definitions of those concepts, based on the definitions in [2].

To start with, the original message Alice wants to send is known as the *plaintext*. The method she uses to 'hide' the original message - in her case, writing it in a secret alphabet - is also called the *cypher* she uses. The secret alphabet she and Bob use is called the *key*. The key is a parameter of the cypher they use, since they could choose any type of secret alphabet they like. The version of the message that is written in the secret alphabet is known as the *cyphertext*.

Furthermore, the process Alice goes through to 'hide' her original message is the *encryption*, and the process Bob goes through when he retrieves the plaintext from the cyphertext is the *decryption*. The exact steps they have to take for encryption and decryption are called the *encryption function* and *decryption function* respectively. In Bob's case the decryption function could be something like this:

1. Take a symbol from the cyphertext.
2. Look up which letter is associated with the symbol, according to the secret alphabet.
3. Replace the symbol by the corresponding letter.
4. Repeat this until every symbol has been changed into a letter.

If Bob follows these steps, he will certainly retrieve the plaintext from the cyphertext, and thus complete the decryption.

Lastly, the whole of all possible keys, encryption functions, decryption functions, plaintexts and cyphertexts for a certain cypher is called a *cryptosystem*.

In the case of Example 1.1.1, Alice and Bob use the same key, namely the secret alphabet, for encrypting and decrypting the message. This is called *symmetric-key cryptography*. On the other hand, it is also possible that Alice and Bob both use a different key, which is called *asymmetric-key cryptography*, or *asymmetric encryption* for short. We will illustrate this type of encryption with another example.

Example 1.1.2. When Bob finds the envelope containing Alice's message, he sees that it has been opened before, probably by Eve. Bob knows Alice wrote her message in the secret alphabet, but he starts to feel a little worried that Eve might still have been able to retrieve the original message. Eve is quite good at puzzles and such like, so she might have figured out the secret alphabet by analysing Alice's encrypted

message. Thus, Bob wants to find a way to make sure that the next time he gets a letter from Alice it really cannot be decrypted by Eve.

Seeing as Bob and Alice still cannot meet up because of the pandemic, they cannot make a new secret alphabet. However, Bob has found another solution. He decides to rent a post office box, which has a combination lock on it and only Bob knows the combination of the lock. He sends Alice the address of the post office box and tells her to send all confidential letters there. Now even if Eve finds out the address of the post office box, she will still not be able to get her hands on Alice's letters, for only Bob knows the combination that opens the post office box. △

In this example, Bob considered the possibility that Eve had figured out the secret alphabet by analysing Alice's message. If this indeed were the case, then Eve would be able to retrieve the plaintext of the message from the cyphertext. Any process that allows a third party who does not possess the key to succesfully retrieve the plaintext from a certain cyphertext, is called an *attack* on the cryptosystem that was used. So in the case of this example, finding the secret alphabet through analysing a cyphertext is an attack. In particular, this attack exposes the key that was used. As we will see later on, an attack does not have to expose the key to be able to find the plaintext.

As we saw in Example 1.1.2 Bob realised that the cryptosystem that he and Alice used could have been attacked, so he decided to use a new cryptosystem for further communication. The post office box can be considered the cypher of this new cryptosystem. As we have seen, Alice and Bob each have a different key. Alice's key is the address of the post office box, while Bob's key is the combination of the lock on the post office box. The key that Bob uses is called the *private key*, seeing as it is only known by one person, Bob, and he keeps it private. The key that Alice uses is called the *public key*. As the name suggests, the public key can be made public and used by anyone who wants to send Bob a confidential letter. The fact that the public key is public does not compromise the security of the encryption that is used, for Bob is still the only person who knows the private key and thus the only person who can open the post office box. Therefore, decryption of a message can only be done by Bob.

In practice, most communication is done via the internet. Even though you could make a digital equivalent of the post office box, a message could still be intercepted on its way there. Therefore, asymmetric encryption is usually used on the message itself, so that the message stays secret even if it is intercepted. In that case, hard mathematical problems are used for asymmetric encryption. One example of such a problem is prime factorisation. When given a set of primes it is quite easy to calculate their product, but when given the product is might be a lot harder to determine its prime factorisation. In asymmetric encryption the primes could then be used as the private key, and the product of the primes as the public key. In that way the private key can not, or at least not easily, be found by a person who knows the public key, which makes the asymmetric encryption safe.

1.2 P vs. NP

As we mentioned in the introduction, the rise of quantum computers poses a threat to cryptosystems. The most important reason for this is the fact that quantum computers enable a whole new kind of computation. This allows for new types of computation processes, which can be used to solve certain hard mathematical problems quite efficiently. In this section we will discuss what it means for a certain mathematical problem to be 'hard to solve' and what we mean by solving a problem 'efficiently'. We will base this discussion on [3, Section 1-2].

We will start with an example.

Example 1.2.1 (Knapsack Problem)**.** Suppose you are going on a hike, and you are packing your knapsack. You have selected some items that you would like to bring, but unfortunately you cannot bring all of them, for the total weight of the items exceeds the maximum weight that the knapsack can carry. In order to decide which items you should pack and which you should leave at home, you have decided to assign each item a 'value-score', represented by a positive integer. For example, you give your water bottle a relatively high score of 100, since it is a necessity when going on a hike, while you give an extra pair of shoes a relatively low score of 5, because you probably will not need those. Knowing the weight and value of the items, you want to pack those items that have the largest possible total value, while their total weight is less than or equal to the maximum weight the knapsack can carry. The problem of determining for which set of items this holds, is known as the Knapsack Problem. △

One way of determining the set of items that gives you the largest possible value while not exceeding the maximum possible weight, would be to compute the value of each possible configuration of items that has a total weight less than or equal to the maximum. However, this process quickly becomes very time consuming when looking at a large number of items. Thus, we would like to know whether there is an efficient process by which we can determine the optimal set of items for any instance of the Knapsack Problem.

A process that gives a solution for any instance of a problem is what we call an *algorithm*. We would now like to know whether there is an 'efficient' algorithm that solves the Knapsack Problem. To discuss this idea of efficiency, we will state the Knapsack Problem in terms of a "yes/no" question, also called a *decision problem*. We will base this statement of the Knapsack Problem on [3, Section 6 of Appendix].

Example 1.2.2 (Knapsack Decision Problem)**.** Suppose we are given a finite set U of items, for each item $u \in U$ a weight $w(u) \in \mathbb{Z}^+$ and a value $v(u) \in \mathbb{Z}^+$, a positive integer M, which is the maximum weight the knapsack can carry, and a positive integer T. We would like to answer the following question:

Is there a subset $V \subseteq U$ such that the total weight of V is less than or equal to the maximum weight, so $\sum_{u \in V} w(u) \leq M$, and the total value of V is greater than or equal to T, so $\sum_{u \in V} v(u) \geq T$?

△

We see that if we can find a solution to the Knapsack Problem for a certain set of items U and a certain M, then we can answer the question in Example 1.2.2 for the same set U and the same M, and for any T. After all, if we know the subset V that has the largest total value while its total weight is less than or equal to M, we just need to compare the value of that set V with the given T to answer the question. Thus, we see that the Knapsack Decision Problem is not harder to solve than the Knapsack Problem.

We will now explain what exactly we mean by an algorithm that solves a problem efficiently, by looking at the Knapsack Decision Problem. First of all, we note that we say an algorithm *solves* a decision problem if it gives a "yes/no" answer for any instance of the problem. Now write $n := |U|$. We will call n the problem size. If the time it takes an algorithm to solve a problem can be expressed as a polynomial in n, then we call that algorithm a *polynomial time* algorithm. A polynomial time algorithm is what we consider an efficient algorithm. Generally, we consider a decision problem to be 'easy' to solve if there exists a polynomial time algorithm that solves it. The following definition describes this idea more formally.

Definition 1.2.3. A decision problem is said to be in **class P** or simply **P** (for "**p**olynomial time") if there exists a polynomial time algorithm that solves the problem.

As it turns out, there is no known polynomial time algorithm that solves the Knapsack Decision Problem. However, if someone happened to guess a subset of items that has total weight less than or equal to M and total value greater than or equal to T, so that he can answer the question in Example 1.2.2 with "yes", then checking whether that answer is correct can be done rather quickly. It is just a matter of calculating the weight and the value of the subset, and comparing those to M and T. To be precise, given the subset that was guessed, verifying the "yes" answer can be done in polynomial time. This gives rise to the definition of another class of decision problems.

Definition 1.2.4. A problem is said to be in **class NP** or simply **NP** if, given a guessed solution, a "yes"-answer can be verified in polynomial time.

Here NP stands for "**n**ondeterministic **p**olynomial time". The "nondeterministic" part comes from the fact that we guess a solution, and not calculate it.

With this definition, we see that the Knapsack Problem is in NP. We also see that P $\subseteq$ NP, for if a problem can be solved in polynomial time, we must be able to verify a "yes" answer in polynomial time as well. It is not known whether or not P = NP, but it is suspected that P $\subsetneq$ NP, since there are many problems in NP like the Knapsack Problem, for which there is no known polynomial time algorithm.

If indeed P $\subsetneq$ NP, then there are problems in NP that can never be solved by a polynomial time algorithm, and can therefore be considered hard to solve. Yet with the use of quantum computation, polynomial time algorithms have been found for a few problems in NP, see [4]. However, this does not have to mean that with quantum computation every problem becomes solvable in polynomial time. To further explain this, we need a new definition.

Definition 1.2.5. A problem is called **NP-complete** if it is in NP and all other problems in NP can be reduced to it.

The Knapsack Decision Problem is an example of an NP-complete problem [3, Appendix Section 6].

As we said, there are problems in NP for which polynomial time quantum algorithms have been found. However, those problems are not NP-complete, which means that their algorithms cannot be used for all problems in NP. Nevertheless, if a polynomial time algorithm were found for an NP-complete problem, then all problems in NP would be solvable in polynomial time, since any problem in NP can be reduced to that NP-complete problem. Thus, we could say that NP-complete problems are the hardest problems in NP. Furthermore, it is expected that even with quantum computation, no polynomial time algorithms will be found for NP-complete problems [1, Section 1]. This makes NP-complete problems ideally suited for use in an asymmetric cryptosystem.

1.3 Algebraic surfaces and the Section Finding Problem

Seeing as the cryptosystem described in this paper is based on algebraic surfaces, we will now briefly discuss this topic. We will base our discussion on [1, Section 2.1].

Let p be a prime and let $\mathbb{F}_p$ be the corresponding finite prime field of p elements. Suppose $n, m \in \mathbb{Z}^+$ and

$$f_1(x_1, ..., x_n) = 0, \ ..., \ f_m(x_1, ..., x_n) = 0,$$

is a set of linearly independent algebraic equations over $\mathbb{F}_p$, that is, all coefficients take value in $\mathbb{F}_p$. When $n-m = 2$, so there are two more variables then there are equations, then we call the set of all possible solutions to the equations f_i an *algebraic surface over* $\mathbb{F}_p$. Note that an algebraic surface is a two-dimensional object, since two variables can be chosen freely and all other variables are fixed by the given equations.

Recall that an algebraic curve over $\mathbb{F}_p$ can be described by a set of linearly independent algebraic equations over $\mathbb{F}_p$ such as

$$g_1(x_1, ..., x_n) = 0, ..., g_m(x_1, ..., x_n) = 0,$$

where $n, m \in \mathbb{Z}^+$ and $n - m = 1$. Now we see that an algebraic surface is the same concept as an algebraic curve, the difference being that an algebraic surface is two-dimensional where an algebraic curve is one-dimensional.

For their cryptosystem, Akiyama and Goto use an algebraic surface in so-called affine 3-space over $\mathbb{F}_p$, denoted by $\mathbb{A}^3_{\mathbb{F}_p}$. Affine 3-space over $\mathbb{F}_p$ is the set of tuples (x_1, x_2, x_3) such that $x_i \in \mathbb{F}_p$ for $i = 1, 2, 3$.

We can also view that space in a different way. Consider $\mathbb{F}_p[x, y, t]$, which is the polynomial ring over $\mathbb{F}_p$ in three variables. If we require that the variables take value in $\mathbb{F}_p$, then the set of all tuples that can occur as a solution to an element of $\mathbb{F}_p[x, y, t]$ is $\mathbb{A}^3_{\mathbb{F}_p}$.

In order to construct an algebraic surface in $\mathbb{A}^3_{\mathbb{F}_p}$, Akiyama and Goto choose one irreducible polynomial $f(x, y, t) \in \mathbb{F}_p[x, y, t]$ and define:

$$X := \{(x_1, x_2, x_3) \in \mathbb{A}^3_{\mathbb{F}_p} : f(x_1, x_2, x_3) = 0\}.$$

Seeing as affine 3-space is three-dimensional and f gives one relation that the tuples need to satisfy, so it fixes one variable in terms of the other two variables, we see that X is indeed two-dimensional and therefore an algebraic surface.

On the surface X we can find many different curves and points. Akiyama and Goto make use of a special kind of curves on X that are generally very hard to find when one only knows the equation $f(x, y, t) = 0$ that defines X [1, pg. 427]. In that way, it is safe to use such a curve as the private key and the surface as the public key.

The special kind of curves that we are talking about are parametrized curves on X, that is, curves of which any point (x, y, t) can be given by

$$(x, y, t) = (u_x(t), u_y(t), t),$$

where $u_x, u_y \in \mathbb{F}_p[t]$. If we define a map $\sigma : X \to \mathbb{F}_p$ by $\sigma(x, y, t) = t$ then the map given by $\tau(t) = (u_x(t), u_y(t), t)$ is an inverse map $\tau : \mathbb{F}_p \to X$ such that

$$(\sigma \circ \tau)(t) = \sigma(u_x(t), u_y(t), t) = t.$$

Thus, we see that $\sigma \circ \tau = \mathrm{id}_{\mathbb{F}_p}$. In this case we call the map σ a *fibration* of X on $\mathbb{F}_p$ and the map τ a *section* of σ. Now that we know these terms, we can define the main problem on which the Algebraic Surface Cryptosystem is based.

Definition 1.3.1 (Section Finding Problem)**.** Let $\mathbb{F}_p$ be a finite prime field, and X an algebraic surface over $\mathbb{F}_p$ given by $X(x, y, t) = 0$. The problem of finding a parametrized curve on X is called a *section finding problem* on X.

We can also look at this problem from a different view. To illustrate this, let us look at an example.

Example 1.3.2. Suppose we are working with $\mathbb{F}_7$. Then the polynomial $f \in \mathbb{F}_7[x, y, t]$ defined by

$$f(x, y, t) = x^2y^2t^2 + 5x^2y^2t + 6x + 2xt + t + 3$$

is irreducible and describes an algebraic surface X over $\mathbb{F}_7$ by the following equation:

$$f(x, y, t) = 0. \tag{1.1}$$

Now suppose $(u_x(t), u_y(t))$ is a section on X. Then we must have that $f(u_x(t), u_y(t), t) = 0$.

If we write f as

$$f(x, y, t) = (t^2 + 5t)x^2y^2 + (2t + 6)x + (t + 3),$$

then we can also look at f as a polynomial $\tilde{f}(x, y)$ with coefficients in $\mathbb{F}_7[t]$, seeing as $(t^2+5t), (2t+6), (t+3) \in \mathbb{F}_7[t]$. We now see that Equation (1.1) is equivalent to:

$$\tilde{f}(x, y) = 0.$$

This is one equation in two variables, so it defines a curve. Thus, we see that Equation (1.1) also describes an algebraic *curve* C over $\mathbb{F}_7[t]$. Furthermore, we have that

$$\tilde{f}(u_x(t), u_y(t)) = f(u_x(t), u_y(t), t) = 0.$$

Thus, we see that the section $(u_x(t), u_y(t))$ on the algebraic surface X is also a rational point on the algebraic curve C. △

As this example shows, the problem of finding a section on an algebraic surface X over $\mathbb{F}_p$ is equivalent to the problem of finding a rational point on a curve C over $\mathbb{F}_p[t]$. In this paper we will therefore describe the Algebraic Surface Cryptosystem in terms of algebraic curves instead of algebraic surfaces.

Finding a rational point on a curve over a finite field is a hard problem. Akiyama and Goto showed that this problem can be reduced to solving a system of multi-variable equations of large degrees over $\mathbb{F}_p$. Even when we only look at $\mathbb{F}_2$, solving such a multi-variable equation system is an NP-complete problem [3, Section 7 of Appendix]. Thus, we see that the Algebraic Surface Cryptosystem is based on an NP-complete problem.

2 The original Algebraic Surface Cryptosystem

In this chapter we will explain the original version of the Algebraic Surface Cryptosystem. We will base our explanation on [5, Section 2].

Firstly, we choose a positive prime p and we let $R = \mathbb{F}_p[t]$ be the polynomial ring over $\mathbb{F}_p$. Furthermore, let $K = \mathbb{F}_p(t)$ be the field of rational functions over $\mathbb{F}_p$. Note that K is the field of fractions of R.

Example. Throughout this chapter we will use a toy example to illustrate our explanation. For this example we choose $p = 2$, so our finite prime field is $\mathbb{F}_2$. In this case an example of a function in R is $t^2 + 1$ and an example of a function in K is $\frac{t}{t^3+t^2}$. △

2.1 Keys

Seeing as ASC is an asymmetric-key cryptosystem, we make a distinction between the private and public key.

Private key. The private key consists of two points $U, V \in R^2$, so $U = (u_x, u_y)$ and $V = (v_x, v_y)$ with $u_x, u_y, v_x, v_y \in R$.

Public key. The public key consists of four elements, namely:

- The prime p that we chose for our finite prime field $\mathbb{F}_p$.
- The equation $X(x, y) = 0$ of an algebraic curve through U and V.
- An integer l, which will be used as a lower bound for the degree of a polynomial $f \in R$ that will be constructed during the encryption.
- An integer d, that is chosen in such a way that $d \geq \max\{\deg_t u_x, \deg_t u_y, \deg_t v_x, \deg_t v_y\}$.

2.2 Key generation

The key generation starts with choosing points $U = (u_x, u_y)$ and $V = (v_x, v_y)$ in R^2. We will do this in the following way:

1. Randomly choose $v_x, v_y, \lambda_x, \lambda_y \in R$, such that $\lambda_x \mid \lambda_y$ and $\deg_t \lambda_y > \deg_t v_y$.
2. Choose u_x and u_y as $u_x = v_x + \lambda_x$ and $u_y = v_y + \lambda_y$.

Notice that by choosing u_x and u_y in this way, we have that $u_x - v_x = \lambda_x$ and $u_y - v_y = \lambda_y$. This implies that $(u_x - v_x) \mid (u_y - v_y)$, which will turn out to be useful when defining a curve through U and V. Also notice that by requiring $\deg_t \lambda_y > \deg_t v_y$, we have that $\deg_t u_y > \deg_t v_y$. This ensures that U and V are linearly independent, and thus a curve going through them cannot be a straight line, which might make it too easy to find rational points of the curve. As a side note, any other condition on the degrees of λ and v that ensures that U and V are linearly independent would work as well.

For the next part of the key generation, we will define a curve through U and V. As we discussed in Section 1.3, we want to describe our curve with a single equation in two variables, which we will denote by $X(x, y) = 0$. Then we see that $X \in R[x, y]$ and therefore X must be of the following form:

$$X(x, y) = \sum_{(i,j)} c_{ij} x^i y^j,$$

where $(i, j) \in \mathbb{N}^2$ and $c_{ij} \in R$ for all (i, j). Moreover, there can only be a finite number of pairs $(i.j)$ such that $c_{ij} \neq 0$. Seeing as X is a curve through U and V we must have that $X(u_x, u_y) = X(v_x, v_y) = 0$. This is equivalent to

$$\sum_{(i,j)} c_{ij} u_x^i u_y^j = \sum_{(i,j)} c_{ij} v_x^i v_y^j = 0. \tag{2.1}$$

From this we get the following expression for c_{00}:

$$c_{00} = -\sum_{(i,j)\neq(0,0)} c_{ij} u_x^i u_y^j = -\sum_{(i,j)\neq(0,0)} c_{ij} v_x^i v_y^j. \tag{2.2}$$

On the other hand, Equation (2.1) also gives us the following equation:

$$\sum_{(i,j)\neq(0,0)} c_{ij}(u_x^i u_y^j - v_x^i v_y^j) = 0.$$

From this equation we can get an expression for c_{10}:

$$c_{10}(u_x - v_x) = -\sum_{(i,j)\neq(0,0),(1,0)} c_{ij}(u_x^i u_y^j - v_x^i v_y^j). \tag{2.3}$$

In addition, we have that

$$u_x^i u_y^j - v_x^i v_y^j = (u_x^i - v_x^i)u_y^j + (u_y^j - v_y^j)v_x^i,$$

and

$$u_x^i - v_x^i = (u_x - v_x)(u_x^{j-1} + u_x^{j-2}v_x + ... + v_x^{j-1});$$
$$u_y^j - v_y^j = (u_y - v_y)(u_y^{j-1} + u_y^{j-2}v_y + ... + v_y^{j-1}).$$

As we stated earlier, we have that $(u_x - v_x) \mid (u_y - v_y)$. We now see that then $(u_x - v_x) \mid u_x^i u_y^j - v_x^i v_y^j$ as well. This means that both sides of Equation (2.3) are divisible by $(u_x - v_x)$. This makes it possible to calculate c_{10} if we know the right-hand side of Equation (2.3).

We can now construct X as follows:

3. For each pair of indices $(i,j) \neq (0,0), (1,0)$, choose a random element $c_{ij} \in R$.
4. Compute c_{10} using Equation (2.3).
5. Compute c_{00} using Equation (2.2).

To conclude this section, we will look at the key generation in our toy example.

Example. We will go through the key generation step-by-step:

1. We choose $v_x = t^2 + t$, $v_y = t + 1$, $\lambda_x = t$ and $\lambda_y = t^3$. We see that then indeed $\lambda_x \mid \lambda_y$ and $\deg_t \lambda_y = 3 > 1 = \deg_t v_y$.
2. We now get $u_x = t^2 + 2t \equiv t^2$, since we are working in $\mathbb{F}_2[t]$, and $u_y = t^3 + t + 1$.
3. We choose $c_{01} = t^2 + 1$ and $c_{ij} = 0$ for all other $(i,j) \neq (0,0), (1,0)$.
4. Plugging these elements into Equation (2.3) we get
$$c_{10}(u_x - v_x) = -c_{01}(u_y - v_y) = -(t^2+1)(t^3) = -(t^5 + t^3) \equiv t^5 + t^3.$$
Dividing both sides by $(u_x - v_x) = t$ we find that $c_{10} = t^4 + t^2$.
5. Lastly, we compute c_{00} using Equation (2.2):
$$\begin{aligned} c_{00} &= c_{10}u_x + c_{01}u_y \\ &= (t^4 + t^2)t^2 + (t^2 + 1)(t^3 + t + 1) \\ &= t^6 + t^5 + t^4 + 2t^3 + t^2 + t + 1 \\ &\equiv t^6 + t^5 + t^4 + t^2 + t + 1. \end{aligned}$$

Thus, we have generated our keys. The elements of our private key are $U = (t^2, t^3+t+1)$ and $V = (t^2+t, t+1)$. The elements of our public key are $p = 2$ and

$$X(x,y) = t^6 + t^5 + t^4 + t^2 + t + 1 + (t^4 + t^2)x + (t^2 + 1)y.$$

Furthermore, for the parameters l and d of the public key we choose $l = 5$ and $d = 3$. We see that then $d = \max\{\deg_t u_x, \deg_t u_y, \deg_t v_x, \deg_t v_y\}$. △

2.3 Encryption

Suppose M is the secret message we want to encrypt. Since we are working in R, we would like to translate our message M to an element of R. To be able to do so, we require that M is expressed as a string of elements of $\mathbb{F}_p$. In that way, we can take the elements of M and use them as the coefficients of a polynomial $m \in R$, which we will call a *message polynomial*. Thus, we can turn a string of k elements into a polynomial of degree $k-1$.

Furthermore, we want any message polynomial $m \in R$ to be of degree smaller than l, to enable decryption. Further explanation of this will be given in the next section. Thus, we need to divide our message M into blocks of at most l elements, turn each block into a message polynomial and encrypt each message polynomial separately. We will now look at the encryption of a message polynomial $m \in R$ with $\deg_t m < l$.

1. Firstly, choose a random polynomial $s(x, y) \in R[x, y]$ that satisfies
$$(\deg_x s + \deg_y s)d + \deg_t s < l. \tag{2.4}$$
The purpose of this condition will become clear during the decryption process.
2. Choose another random polynomial $r(x, y) \in R[x, y]$.
3. Randomly choose a monic irreducible polynomial $f \in R$ such that
$$\deg_t f > l. \tag{2.5}$$
The purpose of this condition will also become clear in the decryption process.
4. Compute the so-called cypher polynomial $F(x, y) \in R[x, y]$ using the following formula:
$$F(x, y) = m + fs(x, y) + X(x, y)r(x, y). \tag{2.6}$$

In this case the cypher polynomial is our cyphertext, so this will be sent to the person knowing the private key.

Example. Suppose the secret message we want to send is '11', so the number eleven, which would be a logical response when asked what the fifth prime number is. Since we are working with $p = 2$, we want to write this message as a string of bits. In this case we can just use the binary representation of 11, which is 1011. Since the number of bits is 4 and we chose $l = 5$, we have that $4 < l$. Thus, we do not have to divide our message into smaller blocks for the encryption.

We now construct a polynomial corresponding to 1011 by using the bit in place i as coefficient of t^i, where we count the places in the string of bits from right to left. This gives the polynomial $m(t) = t^3 + t + 1$, where indeed $\deg m < l$. We are now ready to start the encryption.

1. We choose $s(x, y) = tx + t \in R[x, y]$. Since $l = 5$, we see that indeed
$$(\deg_x s + \deg_y s)d + \deg_t s = (1 + 0)3 + 1 = 3 + 1 = 4 < 5 = l.$$
So Condition (2.4) is satisfied.
2. We choose $r(x, y) = y \in R[x, y]$.
3. Now we choose $f = t^6+t+1$. Then f is indeed monic, for its leading term has coefficient 1. Furthermore, one can check that f is not divisible by any irreducible factor of degree 1, 2 or 3 in $\mathbb{F}_p[t]$, and therefore f is indeed irreducible. Lastly, we have that $\deg f = 6 > l$, so Condition (2.5) is satisfied.
4. Substituting all these polynomials into Formula (2.6), we find
$$\begin{aligned} F(x, y) &= (t^3 + t + 1) + (t^6 + t + 1)(tx + t) + (t^6 + t^5 + t^4 + t^2 + t + 1 + (t^4 + t^2)x + (t^2 + 1)y)y \\ &= (t^4 + t^2)xy + (t^7 + t^2 + t)x + (t^2 + 1)y^2 + (t^6 + t^5 + t^4 + t^2 + t + 1)y + t^7 + t^3 + t^2 + 2t + 1 \\ &\equiv (t^4 + t^2)xy + (t^7 + t^2 + t)x + (t^2 + 1)y^2 + (t^6 + t^5 + t^4 + t^2 + t + 1)y + t^7 + t^3 + t^2 + 1. \end{aligned}$$

Now that we have constructed our cypher polynomial, the encryption is done. △

2.4 Decryption

Once the person who knows the private key receives the encrypted message, they can decrypt it as follows:

1. Evaluate the cypher polynomial F at the points U and V, which are precisely the private key, to get the following polynomials:
$$h_1 := F(u_x, u_v) = m + fs(u_x, u_y);$$
$$h_2 := F(v_x, v_y) = m + fs(v_x, v_y).$$
Notice that the term $X(x,y)r(x,y)$ is absent in both polynomials, seeing as $X(u_x, u_y) = X(v_x, v_y) = 0$. Also note that $h_1, h_2 \in R$.
2. Factor the polynomial $h_1 - h_2 = f(s(u_x, u_y) - s(v_x, v_y))$ into irreducible factors. Then we find f as the factor of the largest degree.
3. Now compute m using $m \equiv h_1 \pmod f$.

We have now retrieved m in the form of a polynomial in R. All that is left to do then is to translate m back to the original string M of elements of $\mathbb{F}_p$, and the decryption is finished.

In step 2 we stated that f must be the largest degree factor of $h_1 - h_2$. For completeness, we will show why that must be the case. Firstly, by the calculation rules for degrees we have that

$$\deg_t s(u_x, u_y) \leq (\deg_x s) \cdot (\deg_t u_x) + (\deg_y s) \cdot (\deg_t u_y) + \deg_t s.$$

Seeing as we chose $d \geq \max\{\deg u_x, \deg u_y, \deg v_x, \deg v_y\}$, we have that $\deg_t u_x \leq d$ and $\deg_t u_x \leq d$. This yields

$$(\deg_x s) \cdot (\deg_t u_x) + (\deg_y s) \cdot (\deg_t u_y) + \deg_t s \leq (\deg_x s + \deg_y s) \cdot d + \deg_t s < l,$$

where the last inequality holds because we chose s such that it satisfies Condition (2.4).

In a similar way we find that $\deg_t s(v_x, v_y) < l$. Therefore we must have that $\deg_t(s(u_x, u_y) - s(v_x, v_y)) < l$, and thus any irreducible factors of $s(u_x, u_y) - s(v_x, v_y)$ must also be of degree strictly smaller than l. On the other hand, we chose f with degree strictly greater than l. Since f is irreducible, it must be the irreducible factor of the largest degree of $f(s(u_x, u_y) - s(v_x, v_y))$. Hence it is certain that we found the right f in step 2.

To conclude this section, we return to a remark we made in the previous section. There we said that we wanted any message polynomial $m \in R$ to be of degree smaller than l, to enable decryption. Suppose we would use a message polynomial m with $\deg_t m \geq l$. Then it could happen that $\deg_t m = \deg_t fs(u_x, u_y)$, seeing as $\deg_t fs(u_x, u_y) \geq \deg_t f > l$. Moreover, we could have that $m = -fs(u_x, u_y)$, which would mean that

$$h_1 = F(u_x, u_y) = m + fs(u_x, u_y) = 0.$$

Thus, in step 2 we would get $h_1 - h_2 = -h_2 = -m - fs(v_x, v_y)$ and then we would not be able to find f as a factor of $h_1 - h_2$ anymore. We now see that we must have that $\deg_t m < l$ in order for the decryption to work.

Example. We will now decrypt the cypher polynomial F to retrieve the original message.

1. Evaluating F at U and V gives
$$h_1 := F(U) = F(t^2, t^3 + t + 1) = (t^7 + t^2 + t)t^2 + t^7 + t^3 + t^2 + 1;$$
$$h_2 := F(V) = F(t^2 + t, t + 1) = (t^7 + t^2 + t)(t^2 + t) + t^7 + t^3 + t^2 + 1.$$
2. We will now factor $h_1 - h_2$:
$$\begin{aligned} h_1 - h_2 &= (t^7 + t^2 + t)t^2 + t^7 + t^3 + t^2 + 1 - ((t^7 + t^2 + t)(t^2 + t) + t^7 + t^3 + t^2 + 1) \\ &= (t^7 + t^2 + t)t^2 - (t^7 + t^2 + 1)(t^2 + t) \\ &= -(t^7 + t^2 + t)t \\ &\equiv t^8 + t^3 + t^2 \\ &= t^2(t^6 + t + 1). \end{aligned}$$

The factor with the largest degree is $t^6 + t + 1$, so we must have that $f = t^6 + t + 1$.

3. We now find
$$h_1 = (t^6 + t + 1)(t^3 + t) + t^3 + t + 1 \equiv t^3 + t + 1 \pmod{f}.$$
So $m = t^3 + t + 1$.

We can now turn m back into a string of bits by putting the coefficient of t^i in place i of the string of bits, once again counting from right to left. This gives 1011, which is the binary representation of the number 11, and so we have retrieved the original message. △

3 An attack on the Algebraic Surface Cryptosystem

As it turns out, the Algebraic Surface Cryptosystem as described in Chapter 2 is vulnerable to different attacks. One such attack was devised by Ivanov and Voloch, as described in [5, Section 3]. An important thing to note is that their attack does not solve the very hard problem of finding rational points on the curve X, and therefore the private key remains secret. The idea of their attack however is to try and decypher the encrypted message without any knowledge of the private key. They do this by working in an extension of the polynomial ring R in which they can find a point on the curve X, and then use those points to evaluate the cypher polynomial. We will explain how this attack works exactly, where we base our explanation on [5, Section 3]. Subsequently, we will discuss what changes in the ASC could make it resistant to this attack, based on the discussion in [1, Section 3.5].

3.1 The attack

Firstly, recall that we denoted $R = \mathbb{F}_p[t]$, the ring of polynomials over $\mathbb{F}_p$, and $K = \mathbb{F}_p(t)$, the field of rational functions over $\mathbb{F}_p$. Note that $R \subset K$.

We define $S = R[x]/(X(x,0))$. Notice that indeed $X(x,0) \in R[x]$, since $X(x,y)$ is a polynomial in two variables over R, and by choosing $y = 0$ we are left with a polynomial in x over R.

Let π be the natural projection

$$\begin{aligned} \pi : R[x] &\to S, \\ f &\mapsto f \pmod{X(x,0)}. \end{aligned}$$

If we write $\alpha := \pi(x) \in S$, then we have that

$$X(\alpha, 0) = X(\pi(x), 0) = \pi(X(x,0)) = 0.$$

Thus we see that α is a point on the curve $X(x,0) = 0$ over S. Moreover, we have that

$$S = R[x] \Big/ (X(x,0)) \simeq R[\alpha].$$

So we see that S is just R extended with a root of $X(x,0)$, namely α.

From now on we will consider polynomials over R as polynomials over $S \simeq R[\alpha]$. We then evaluate our cypher polynomial F at $(\alpha, 0)$ to get

$$F(\alpha,0) = m + fs(\alpha,0) + r(\alpha,0)X(\alpha,0) = m + fs(\alpha,0). \tag{3.1}$$

We now want to find f, using Equation (3.1). Seeing as $f \in R$, we want to go from S back to R. In order to do so, we define $L = S \otimes_R K$, which is the so-called *tensor product* of S and K over R. Since S is an R-module, this tensor product L is a K-module. For further explanation of this and the tensor product in general, we refer the reader to [6, Section 10.4]. In essence, we have that

$$L = K[x] \Big/ (X(x,0)).$$

Seeing as K is a field an L is a K-module we can consider L as a K-vectorspace of degree $[L : K] \in \mathbb{Z}^+$. Furthermore, we know that $R \subset K$ and thus $R[x] \subset K[x]$, so it follows that $S \subset L$. As it turns out, there is a map from L to K that maps S to R. We will now explain how this map works and why it maps S to R.

The map we use to go from L to K is the trace operator $\mathrm{Tr} : L \to K$. As said before, we can consider L as a K-vectorspace, so after choosing an appropriate basis we can write each element of L as a matrix with elements in K. Then the trace operator $\mathrm{Tr} : L \to K$ is equal to the matrix trace operator. Therefore the trace operator is a K-linear map , which means that for any $A, B \in L$ and any $c \in K$ we have

(a) $\mathrm{Tr}(A + B) = \mathrm{Tr}(A) + \mathrm{Tr}(B)$;

(b) $\mathrm{Tr}(cA) = c\mathrm{Tr}(A)$.

Furthermore, we know that the trace of the identity matrix $\mathbf{1}$ is $\mathrm{Tr}(\mathbf{1}) = [L : K]$. In combination with Property (b) of the trace it follows that for any $c \in K$ we have

$$\mathrm{Tr}(c) = c\mathrm{Tr}(\mathbf{1}) = [L : K]c.$$

Thus we see that the trace operator satisfies $\mathrm{Tr}|_K = [L : K]id$.

We will now look at the image of S under the trace operator. Suppose $g \in S$. Then we know $g \in R[x]$ as well, and thus all coefficients of g are in R. This means that if we write g as a matrix according to the basis we chose for L, each element of the matrix will also be in R. Thus, the trace of g is the sum of elements of R, and therefore $\mathrm{Tr}(g) \in R$. We now see that indeed S gets mapped to R under the trace operator.

For the next part of the attack we want to choose an element $\beta \in S$ such that $\beta \neq 0$ and $\mathrm{Tr}(\beta) = 0$. We write $n := [L : K]$ and for simplicity we assume that $\gcd(p, n) = 1$. We choose a $\gamma \in S$ such that $\gamma \notin R$ and define

$$\beta := \gamma - \frac{\mathrm{Tr}(\gamma)}{n}. \tag{3.2}$$

Seeing as $\gamma \in S\backslash R$ we have that $\gamma \notin K$ and thus $\mathrm{Tr}(\gamma) \neq n\gamma$. It then follows that $\frac{\mathrm{Tr}(\gamma)}{n} \neq \gamma$, hence $\beta \neq 0$. Furthermore, we have that

$$\mathrm{Tr}(\beta) = \mathrm{Tr}\left(\gamma - \frac{\mathrm{Tr}(\gamma)}{n}\right) = \mathrm{Tr}(\gamma) - \mathrm{Tr}(\gamma)\mathrm{Tr}(\frac{1}{n}) = \mathrm{Tr}(\gamma) - \mathrm{Tr}(\gamma)\left(n \cdot \frac{1}{n}\right) = \mathrm{Tr}(\gamma) - \mathrm{Tr}(\gamma) = 0.$$

Here the fact that $\mathrm{Tr}(\frac{1}{n}) = n \cdot \frac{1}{n}$ follows from the fact that $n \in R$ and thus $\frac{1}{n} \in K$, and we know that $\mathrm{Tr}|_K = n \cdot id$.

We have now found a $\beta \in S$ that satisfies our conditions. Using Equation (3.1) we find

$$\mathrm{Tr}(\beta F(\alpha, 0)) = \mathrm{Tr}(\beta m + \beta f s(\alpha, 0)) = \mathrm{Tr}(\beta m) + \mathrm{Tr}(\beta f s(\alpha, 0)).$$

Seeing as $f, m \in R$ we have that $f, m \in K$ as well and therefore

$$\mathrm{Tr}(\beta m) + \mathrm{Tr}(\beta f s(\alpha, 0)) = m\mathrm{Tr}(\beta) + f\mathrm{Tr}(\beta s(\alpha, 0)) = m \cdot 0 + f\mathrm{Tr}(\beta s(\alpha, 0)) = f\mathrm{Tr}(\beta s(\alpha, 0)). \tag{3.3}$$

We will write $p_\beta := f\mathrm{Tr}(\beta s(\alpha, 0))$. Since $\beta, s(\alpha, 0) \in S$ we have that $\beta s(\alpha, 0) \in S$ and therefore $\mathrm{Tr}(\beta s(\alpha, 0)) \in R$. Furthermore, we know that $f \in R$ and thus we see that p_β is a product of elements of R and therefore $p_\beta \in R$ as well. Moreover, by its definition p_β is divisible by f and we know f is monic and irreducible. Hence, if we factor p_β into irreducible factors, one of those factors must be f. This provides us with the following way of finding f:

1. Calculate p_β for several different choices of β.
2. Determine the greatest common divisor of the p_β.
3. Factor the greatest common divisor into irreducible factors.
4. Find f as the irreducible factor of the largest degree.

One idea for the choices of β in step 1 is to use Equation (3.2) and choose different powers of α for γ.

An important thing to remark here is that it could happen that there is more than one irreducible factor of largest degree in step 4. If that is the case, we should calculate p_β for some more choices of β and redo the process from step 2. We should keep doing this until there is only one factor of largest degree in step 4, for only then can we be certain that we found the right f.

Once we have found f, we can find m as follows. We calculate

$$\mathrm{Tr}(F(\alpha, 0)) = \mathrm{Tr}(m) + \mathrm{Tr}(f s(\alpha, 0)) = nm + f\mathrm{Tr}(s(\alpha, 0)), \tag{3.4}$$

where the last step follows because of the fact that $f, m \in K$, as we have seen before.

Seeing as m and f were chosen such that $\deg_t m < l < \deg_t f$, we will find nm as the remainder when we divide $\mathrm{Tr}(F(\alpha, 0))$ by f. Since n is known to us, we can just divide nm by n to find m. Thus, we have succesfully obtained the plaintext.

3.2 Ideas to resist the attack

Akiyama and Goto had two ideas that could make their cryptosystem resistant against the attack we just described. One idea is to make the trace computation very time-consuming, so that it takes too much time to perform the attack. The other idea is to change the form of m and f, so that they cannot be found using the attack.

Both ideas can be realised by making m and f multi-variable. Suppose we replace m and f, which are elements of R, by $m(x)$ and $f(x)$, which are elements of $R[x]$. If we assume that $m(x)$ and $f(x)$ are not constant, then we see that $m(x), f(x) \in R[x] \backslash R$.

Now the calculation of $\mathrm{Tr}(\beta F(\alpha, 0))$ goes as follows:

$$\mathrm{Tr}(\beta F(\alpha,0)) = \mathrm{Tr}(\beta m(\alpha) + \beta f(\alpha) s(\alpha,0)) = \mathrm{Tr}(\beta m(\alpha)) + \mathrm{Tr}(\beta f(\alpha) s(\alpha,0)). \tag{3.5}$$

Seeing as $\alpha \notin K$, we have that $f(\alpha), m(\alpha) \notin K$. This means that we cannot use the linearity of the trace operator to extract the terms $m(\alpha)$ and $f(\alpha)$ from the trace. Therefore (3.3) does not hold when we replace m and f by $m(\alpha)$ and $f(\alpha)$. This makes the computation of $\mathrm{Tr}(\beta F(\alpha, 0)$ more complicated and thus more time-consuming.

Moreover, if Equation (3.3) does not hold, then we cannot find $f(\alpha)$ using the process that was described in the attack. This also means that we cannot find $m(\alpha)$ using the process described in the attack, especially since Equation (3.4) does not hold when we replace m and f by $m(\alpha)$ and $f(\alpha)$. The latter is true by the same reasoning we used before, that is, since $m(\alpha), f(\alpha) \notin K$ we cannot use the linearity of the trace to extract those terms from the trace.

We see that the attack does not properly work anymore when we replace m and f by multi-variable polynomials. In particular, Akiyama and Goto found that their cryptosystem is safe when they replace m and f by $m(x, y), f(x, y) \in R[x, y]$.

4 The improved Algebraic Surface Cryptosystem

To withstand the attack by Ivanov and Voloch and other attacks, Akiyama and Goto, together with Miyake, came up with an improved version of the Algebraic Surface Cryptosystem. This version is described in [1]. In this version, the polynomials $m, f \in R$ are replaced by multi-variable polynomials $m(x,y), f(x,y) \in R[x,y]$. By doing so, they make sure the attack by Ivanov and Voloch cannot be used anymore, as we explained in Section 3.2.

However, the idea of replacing m and f by multi-variable polynomials does lead to a problem in step 2 of the decryption process of the original cryptosystem (see Section 2.4). Seeing as it might happen that $m(u_x, u_y) \neq m(v_x, v_y)$ and $f(u_x, u_y) \neq f(v_x, v_y)$, it becomes uncertain whether $h_1 - h_2 = f(s(u_x, u_y) - s(v_x, v_y))$ when we make m and f multi-variable. Thus, we might not be able to find f anymore, which would disable decryption.

To avoid this problem, Akiyama and Goto decided to use only one point U as the private key and construct two cypher polynomials during the encryption, instead of using two points and one cypher polynomial. In this chapter we will explain this new version of the cryptosystem, where we base our explanation on [1, Section 4].

To start with, we choose a positive prime p. Like we did in Chapter 2, we define $R = \mathbb{F}_p[t]$, which is the ring of polynomials over $\mathbb{F}_p$, and $K = \mathbb{F}_p(t)$, which is the field of rationial functions over $\mathbb{F}_p$.

Example. Throughout this chapter we will use a toy example to illustrate our explanation, just as we did in Chapter 2. For this toy example we also choose $p = 2$, so we are working with $R = \mathbb{F}_2[t]$ and $K = \mathbb{F}_2(t)$. $\triangle$

4.1 Keys

Private key. The private key consists of one point $U = (u_x, u_y) \in R^2$.

Public key. The public key consists of the following elements:

- The prime p that we chose for our finite prime field $\mathbb{F}_p$.
- A polynomial $X(x,y)$ that defines an algebraic curve through U by $X(x,y) = 0$.
- Two sets Λ_m and Λ_f of index pairs $(i,j) \in \mathbb{N}^2$.
- Two sets D_m and D_f of elements of $\mathbb{N}$.

The sets Λ_m, Λ_f, D_m and D_f will be used for the construction of the polynomials m and f during the encryption.

4.2 Key generation

The first step in the key generation is to choose a random point $U = (u_x, u_y)$ in R^2. Then we can construct an algebraic curve X through U. We know that the curve X has to satisfy $X(u_x, u_y) = 0$. As we have seen in Section 2.2, X must be of the form

$$X(x,y) = \sum_{(i,j)} c_{ij} x^i y^j,$$

where $(i,j) \in \mathbb{N}^2$ and $c_{ij} \in R$ for all (i,j). Furthermore, there can only be a finite number of pairs (i,j) such that $c_{ij} \neq 0$. This means that the c_{ij} have to satisfy

$$\sum_{(i,j)} c_{ij} u_x^i u_y^j = 0.$$

From this equation we find the following formula for c_{00}:

$$c_{00} = - \sum_{(i,j) \neq (0,0)} c_{ij} u_x^i u_y^j. \tag{4.1}$$

Knowing this, we can construct X as follows:

1. Randomly choose $I_X, J_X \in \mathbb{Z}^+$.
2. Choose an element $c_{I_X J_X} \in R$ such that $c_{I_X J_X} \neq 0$.
3. For each pair $(i,j) \neq (0,0)$ such that $i \leq I_X$ and $j \leq J_X$ choose a random element $c_{ij} \in R$.
4. For all other $(i,j) \neq (0,0)$ choose $c_{ij} = 0$.
5. Compute c_{00} using Equation (4.1).

Notice that the choices we made for the c_{ij} in step 2 to 4 imply that $\deg_x X = I_X$ and $\deg_y X = J_X$.

For the next part of the key generation, we will choose the sets Λ_f and D_f. We start with Λ_f:

6. Choose $I_f, J_f \in \mathbb{Z}^+$ such that
$$I_f > I_X + 1 \text{ and } J_f > J_X + 1. \tag{4.2}$$
7. Randomly choose a number $n_f \in \mathbb{N}$.
8. For $1 \leq k \leq n_f$ choose pairs $(i_k, j_k) \in \mathbb{N}^2$ such that $i_k \leq I_f$ and $j_k \leq J_f$.
9. Define Λ_f by
$$\Lambda_f := \{(i_k, j_k) \mid 1 \leq k \leq n_f\} \cup \{(0,0); (I_f, J_f); (I_f - I_X - 1, J_f - J_X - 1)\}. \tag{4.3}$$

Now that we know the elements in Λ_f, we construct D_f as follows:

10. Choose $d_{I_f J_f} \in \mathbb{Z}^+$ such that
$$d_{I_f J_f} > \deg_t X + 1. \tag{4.4}$$
11. For all $(i,j) \in \Lambda_f$ with $(i,j) \neq (I_f, J_f)$ choose $d_{ij} \in \mathbb{N}$ such that $d_{ij} \leq d_{I_f J_f}$.
12. Define $D_f := \{d_{ij} \mid (i,j) \in \Lambda_f\}$.

For the final part of the key generation we will choose the sets Λ_m and D_m. We define

$$\Lambda_X := \{(i,j) \mid i,j \in \mathbb{Z}_{\geq 0},\, c_{ij} \neq 0\};$$
$$\Lambda_f \Lambda_X := \{(i_f + i_X, j_f + j_X) \mid (i_f, j_f) \in \Lambda_f,\, (i_X, j_X) \in \Lambda_X\}.$$

We choose Λ_m and D_m as follows:

13. Choose $(I_m, J_m) \in \Lambda_f \Lambda_X$ as
$$(I_m, J_m) = (I_f - 1, J_f - 1). \tag{4.5}$$
14. Choose a random number $n_m \in \mathbb{N}$.
15. For $1 \leq k \leq n_m$ choose different pairs $(i'_k, j'_k) \in \Lambda_f \Lambda_X$ such that $i'_k \leq I_m$ and $j'_k \leq J_m$.
16. Define $\Lambda_m := \{(i'_k, j'_k) \mid 1 \leq k \leq n_m\} \cup \{(0,0); (I_m, J_m)\}$.
17. Choose $d'_{I_m J_m} \in \mathbb{Z}^+$ such that $\deg_t X < d'_{I_m J_m} < d_{I_f J_f}$.
18. For all $(i,j) \in \Lambda_m$ such that $(i,j) \neq (I_m, J_m)$, choose $d'_{ij} \in \mathbb{N}$ such that $d'_{ij} \leq d'_{I_m J_m}$.
19. Define $D_m := \{d'_{ij} \mid (i,j) \in \Lambda_m\}$.

Notice that in step 13 we indeed have that $(I_m, J_m) \in \Lambda_f \Lambda_X$, because

$$(I_f - 1, J_f - 1) = (I_f - I_X - 1 + I_X, J_f - J_X - 1 + J_x),$$

and we know that $(I_X, J_X) \in \Lambda_X$ and $(I_f - I_X - 1, J_f - J_X - 1) \in \Lambda_f$. Now we see that the way we chose Λ_m implies that $\Lambda_m \subset \Lambda_f \Lambda_X$.

Example. We choose $U = (t^2 + 1, t + 1) \in R^2$. We now construct X as follows:

1. We choose $I_X = 1$ and $J_X = 1$.
2. We choose $c_{11} = t$.
3. Now we choose $c_{10} = 0$ and $c_{01} = t^2 + t$.
4. Lastly we choose $c_{ij} = 0$ for all $(i, j) \notin \{(0,0); (1,0); (0,1); (1,1)\}$.
5. Using Equation (4.1) we now compute

$$\begin{aligned} c_{00} &= -(c_{01}u_y + c_{11}u_x u_y) \\ &= -((t^2+t)(t+1) + t(t^2+1)(t+1)) \\ &= -(t^4 + 2t^3 + t^2 + 2t) \\ &\equiv t^4 + t^2. \end{aligned}$$

Thus, we have found

$$X(x, y) = txy + (t^2 + t)y + t^4 + t^2.$$

We will now construct $\Lambda_f, D_f, \Lambda_m$ and D_m:

6. We choose $I_f = 3$ and $J_f = 3$, then we see that Condition (4.2) is indeed satisfied.
7. We now choose $n_f = 1$.
8. We choose $(i_1, j_1) = (1, 1)$, then we see that indeed $i_1 \leq I_f$ and $J_1 \leq J_f$.
9. We now define Λ_f according to Equation (4.3) as

$$\Lambda_f = \{(1,1)\} \cup \{(0,0); (3,3); (1,1)\} = \{(0,0); (1,1); (3,3)\}.$$

10. We choose $d_{33} = 6$, then we see that Condition (4.4) is satisfied.
11. We now choose $d_{00} = 2$ and $d_{11} = 1$.
12. We define $D_f = \{d_{00} = 2, d_{11} = 1, d_{33} = 6\}$.
13. We choose I_m and J_m according to Equation (4.5), so $(I_m, J_m) = (2, 2)$.
14. We choose $n_m = 0$.
15. Seeing as we chose $n_m = 0$, we do not choose any pairs (i'_k, j'_k).
16. We define

$$\Lambda_m = \emptyset \cup \{(0,0); (2,2)\} = \{(0,0); (2,2)\}.$$

17. We choose $d'_{22} = 5$.
18. We choose $d'_{00} = 1$.
19. Lastly we define $D_m = \{d'_{00} = 1, d'_{22} = 5\}$.

To summarize, our private key is $U = (t^2 + 1, t + 1)$ and our public key consists of the following elements:

- $p = 2$.
- $X(x, y) = txy + (t^2 + t)y + t^4 + t^2$.
- $\Lambda_m = \{(0,0); (2,2)\}$ and $\Lambda_f = \{(0,0); (1,1); (3,3)\}$.
- $D_m = \{d'_{00} = 1, d'_{22} = 5\}$ and $D_f = \{d_{00} = 2, d_{11} = 1, d_{33} = 6\}$.

△

4.3 Encryption

Now suppose M is the message we want to encrypt. As we discussed in Chapter 2, we would like to translate our message M to an element of $R = \mathbb{F}_p[x]$, since we are working in R. Therefore we require that M is expressed as a string of elements of $\mathbb{F}_p$. Then we can make a polynomial in R with the $k-th$ element of M as coefficient of the term t^k. In that way a string of n elements becomes a polynomial of degree $n-1$.

Firstly, we divide M into blocks $M_{00}||...||M_{ij}||...||M_{I_m J_m} = M$, where $(i,j) \in \Lambda_m$, such that

$$|M_{ij}| \leq (p-1)(d'_{ij}+1) \text{ for all } (i,j) \in \Lambda_m. \tag{4.6}$$

Here we use $|M_{ij}|$ to denote the number of elements in the string M_{ij}. If the message is too long to have all the blocks M_{ij} satisfy Condition (4.6), then the message should be divided into smaller parts, and each part should go through the encryption process separately.

Subsequently, we divide each M_{ij} into $d'_{ij}+1$ blocks:

$$M_{ij} = M_{ij0}||M_{ij1}||...||M_{ijd'_{ij}}.$$

We will now use these M_{ijk} to construct our message polynomial m:

1. For each $(i,j) \in \Lambda_m$ define
$$m_{ij} = \sum_{k=0}^{d'_{ij}} M_{ijk} t^k. \tag{4.7}$$
2. Define the message polynomial m by
$$m(x,y) = \sum_{(i,j)\in\Lambda_m} m_{ij} x^i y^j. \tag{4.8}$$

Notice that by the way we defined our m_{ij} we have that $m_{ij} \in R$ for each $(i,j) \in \Lambda_m$, and thus we have that $m \in R[x,y]$.

For the next part of the encryption, we need a general definition of Λ_A for a polynomial $A(x,y)$. If we write $A(x,y) = \sum_{(i,j)\in\mathbb{N}^2} a_{ij} x^i y^j$, then we define

$$\Lambda_A := \{(i,j) \in \mathbb{N}^2 \mid a_{ij} \neq 0\}.$$

Notice that the definition of Λ_X in the previous section satisfies this definition. Furthermore, we see that Λ_m also satisfies this definition, because of the way we defined m.

We will now construct our cypher polynomials:

3. Choose an irreducible polynomial $f \in R[x,y]$ such that it satisfies the following conditions:
   (a) Λ_f is as we defined it in (4.3).
   (b) Denote the coefficient of $x^i y^j$ by f_{ij}, which is an element of R. Then $\deg_t f_{ij} = d_{ij}$ for all $(i.j) \in \Lambda_f$, where the d_{ij} are the elements of D_f.
4. Choose two random polynomials $r_1, r_2 \in R[x,y]$ that have the same form as f, that is, these polynomials satisfy
   (a) $\Lambda_{r_1} = \Lambda_{r_2} = \Lambda_f$.
   (b) Let $r_{1,ij}$ and $r_{2,ij}$ denote the coefficients of $x^i y^j$ in r_1 and r_2 respectively. Then $\deg_t r_{1,ij} = \deg_t r_{2,ij} = d_{ij}$ for all $(i,j) \in \Lambda_f$.
5. Choose two random polynomials $s_1, s_2 \in R[x,y]$ that have the same form as X, so they must satisfy
   (a) $\Lambda_{s_1} = \Lambda_{s_2} = \Lambda_X$.
   (b) Let $s_{1,ij}$ and $s_{2,ij}$ denote the coefficients of $x^i y^j$ in s_1 and s_2 respectively. Then $\deg_t s_{1,ij} = \deg_t s_{2,ij} = \deg_t c_{ij}$ for all $(i,j) \in \Lambda_X$.

6. Construct the two cypher polynomials as follows:

$$F_1(x,y) = m(x,y) + f(x,y)s_1(x,y) + X(x,y)r_1(x,y); \tag{4.9}$$
$$F_2(x,y) = m(x,y) + f(x,y)s_2(x,y) + X(x,y)r_2(x,y). \tag{4.10}$$

We are now done with the encryption.

Example. Suppose our message is $M = 11010101$. We divide this into the blocks $M_{00} = 11$ and $M_{22} = 010101$. Recall that $d'_{00} = 1$ and $d'_{22} = 5$, then we see that

$$|M_{00}| = 2 = 1 \cdot 2 = (p-1)(d'_{00}+1);$$
$$|M_{22}| = 6 = 1 \cdot 6 = (p-1)(d'_{22}+1).$$

So Condition (4.6) is satisfied. We now divide M_{00} into $d'_0 0 + 1 = 2$ blocks and M_{22} into $d'_2 2 + 1 = 6$ blocks, which yiels

$$M_{000} = 1,\ M_{001} = 1;$$
$$M_{220} = 0,\ M_{221} = 1,\ M_{222} = 0,\ M_{223} = 1,\ M_{224} = 0,\ M_{225} = 1.$$

We are now ready to go through the encryption steps.

1. We define m_{00} and m_{22} according to Equation (4.7), which yields

$$m_{00} = 1 + t;$$
$$m_{22} = t + t^3 + t^5.$$

2. We now construct the message polynomial $m(x,y)$ according to Equation (4.8), which gives us

$$m(x,y) = (t^5 + t^3 + t)x^2y^2 + t + 1.$$

3. We choose a polynomial $f \in R[x,y]$ as

$$f(x,y) = (t^6+1)x^3y^3 + txy + t^2 + t.$$

Recall that we defined $\Lambda_f = \{(0,0);(1,1);(3,3)\}$ and $d_{00} = 2, d_{11} = 1$ and $d_{33} = 6$. Then we see that f satisfies the conditions (a) and (b) given in step 3. Furthermore, we can check that f is not divisible by any term other than 1, and thus f is irreducible.

4. We choose $r_1, r_2 \in R[x,y]$ according to the conditions given in step 4:

$$r_1(x,y) = t^6x^3y^3 + (t+1)xy + t^2 + 1;$$
$$r_2(x,y) = (t^6+t)x^3y^3 + tcy + t^2.$$

5. We choose $s_1, s_2 \in R[x,y]$:

$$s_1(x,y) = (t+1)xy + t^2y + t^4 + t^3;$$
$$s_2(x,y) = txy + (t^2+1)y + t^4 + 1.$$

Because of the way we defined X we have that $\Lambda_X = \{(0,0);(0,1);(1,1)\}$ and $\deg_t c_{00} = 4, \deg_t c_{01} = 2$ and $\deg_t c_{11} = 1$. We now see that s_1 and s_2 satisfy the conditions given in step 5.

6. We now construct the cypher polynomials F_1 and F_2 according to Formula (4.9) and (4.10), which yields

$$\begin{aligned} F_1(x,y) =& (t^6+t+1)x^4y^4 + (t^7+t^2)x^3y^4 + (t^9+t^8+t^4+t^3)x^3y^3 \\ &+ (t^5+t^3+t)x^2y^2 + txy^2 + (t^3+t^2)xy + (t^2+1)y + t^4 + t^2 + t + 1; \\ F_2(x,y) =& (t^2+t)x^4y^4 + (t^7+t^6+t^3+1)x^3y^4 + (t^8+t^6+t^5+t^4+t^3+1)x^3y^3 \\ &+ (t^5+t^3+t)x^2y^2 + (t^2+t)xy^2 + txy + (t^2+t)y + t^5 + t^4 + t^2 + 1. \end{aligned}$$

The encryption has now been completed, and we can send the cypher polynomials to the person who possesses the private key. △

4.4 Decryption

The person who knows the private key can decrypt the encrypted message as follows:

1. Substitute the private key U into F_1 and F_2 and define

$$h_1 := F_1(u_x, u_y) = m(u_x, u_y) + f(u_x, u_y)s_1(u_x, u_y);$$
$$h_2 := F_2(u_x, u_y) = m(u_x, u_y) + f(u_x, u_y)s_2(u_x, u_y).$$

   We note that the terms containing X are absent in both polynomials, due to the fact that $X(u_x, u_y) = 0$. Furthermore, we see that $h_1, h_2 \in R$, since $u_x, u_y \in R$.

2. Factor the polynomial $h_1 - h_2 = f(u_x, u_y)(s_1(u_x, u_y) - s_2(u_x, u_y))$ into irreducible factors.

3. Calculate $\deg_t f(u_x, u_y)$ as follows:

$$\deg_t f(u_x, u_y) = I_f \cdot (\deg_t u_x) + J_f \cdot (\deg_t u_y) + d_{I_f J_f}. \tag{4.11}$$

4. Find $f(u_x, u_y)$ as a factor of $h_1 - h_2$ whose degree matches $\deg_t f(u_x, u_y)$. This factor does not necessarily have to be irreducible.

5. Calculate $m(u_x, u_y) \equiv h_1 \pmod{f(u_x, u_y)}$.

6. Construct linear equations for the M_{ijk} by comparing the coefficients of t^k on both sides of the equation:

$$m(u_x, u_y) = \sum_{(i,j)\in\Lambda_m} \left(u_x^i u_y^j \sum_{k=0}^{d'_{ij}} M_{ijk} t^k \right). \tag{4.12}$$

7. Solve the equations obtained in step 6 to find the M_{ijk} and reconstruct M as

$$M = M_{000}||...||M_{00d'_{00}}||...||M_{ijk}||...||M_{I_m J_m d'_{I_m J_m}}.$$

In step 4 it might happen that there are multiple factors whose degree matches $\deg_t f(u_x, u_y)$. If the receiver happens to pick a factor that is unequal to f, he will probably not find the right message after decryption. This poses a potential problem when using ASC. However, when using encrypted communication one can send a so-called *Message Authentication Code*, or *MAC* for short, along with a message. After the decryption, the receiver can make certain he found the right message by authenticating this MAC. If the MAC is incorrect, the receiver must have made the wrong choice during step 4. In that case the receiver should return to step 4 and look for another factor whose degree matches $\deg_t f(u_x, u_y)$, and redo the subsequent steps. The receiver should repeat this process until the MAC is authenticated. For further information about the MAC we refer the reader to [7, Section 11.3].

Example. After receiving the cypher polynomials, we are ready to start the decryption process.

1. We substitute $U = (t^2 + 1, t + 1)$ into $F_1(x, y)$ and $F_2(x, y)$ to get

$$h_1 := F_1(t^2 + 1, t + 1) = m(t^2 + 1, t + 1) + f(t^2 + 1, t + 1)s_1(t^2 + 1, t + 1);$$
$$h_2 := F_2(t^2 + 1, t + 1) = m(t^2 + 1, t + 1) + f(t^2 + 1, t + 1)s_2(t^2 + 1, t + 1).$$

2. We now factor $h_1 - h_2$ which yields

$$\begin{aligned} h_1 - h_2 &= f(t^2 + 1, t + 1)(s_1(t^2 + 1, t + 1) - s_2(t^2 + 1, t + 1)) \\ &\equiv t^{17} + t^{16} + t^{15} + t^{14} + t^{11} + t^{10} + t^7 + t^6 + t^3 + t^2 + t + 1 \\ &= (t + 1)^{13}(t^2 + t + 1)^2. \end{aligned}$$

3. We calculate $\deg_t f(t^2+1, t+1)$ using Formula (4.11):

$$\deg_t f(t^2+1, t+1) = 3 \cdot 2 + 3 \cdot 1 + 6 = 15.$$

4. We see that there are multiple factors of $h_1 - h_2$ whose degree is 15. We choose $f(t^2+1, t+1) = (t+1)^{11}(t^2+t+1)^2$.

5. Using the fact that $m(t^2+1, t+1) \equiv h_1 \pmod{f(t^2+1, t+1)}$ we find that

$$m(t^2+1, t+1) \equiv t^1 1 + t^7 + t^5 + 1.$$

6. Substituting our choices into Equation (4.12) gives

$$t^{11}+t^7+t^5+1 = M_{000}+M_{001}t+(t^6+t^4+t^2+1)(M_{220}+M_{221}t+M_{222}t^2+M_{223}t^3+M_{224}t^4+M_{225}t^5).$$

Comparing both sides of this equation gives us the following set of linear equations for the M_{ijk}:

$$\begin{cases} M_{000} + M_{220} & = 1 \\ M_{001} + M_{221} & = 0 \\ M_{222} + M_{220} & = 0 \\ M_{223} + M_{221} & = 0 \\ M_{224} + M_{222} + M_{220} & = 0 \\ M_{225} + M_{223} + M_{221} & = 1 \\ M_{224} + M_{222} & = 0 \\ M_{225} + M_{223} & = 1 \\ M_{224} & = 0 \\ M_{225} & = 1. \end{cases}$$

7. Solving the equation from step 6 we find the following values for the M_{ijk}:

$$M_{000} = 1,\ M_{001} = 1;$$
$$M_{220} = 0,\ M_{221} = 1,\ M_{222} = 0,\ M_{223} = 1,\ M_{224} = 0,\ M_{225} = 1.$$

We now find the original message M as

$$M = M_{000}||M_{001}||M_{220}||M_{221}||M_{222}||M_{223}||M_{224}||M_{225} = 11010101.$$

We have correctly retrieved the original message, and thus the decryption is done. △

References

[1] Koichiro Akiyama, Yasuhiro Goto, and Hideyuki Miyake. "An algebraic surface cryptosystem". In: *Public key cryptography—PKC 2009*. Vol. 5443. Lecture Notes in Comput. Sci. Springer, Berlin, 2009, pp. 425–442. DOI: 10.1007/978-3-642-00468-1_24. URL: https://doi.org/10.1007/978-3-642-00468-1_24.

[2] Brian Beckett. *Introduction to cryptology*. Blackwell Scientific Publications, 1988. ISBN: 978-0-632-01836-9.

[3] Michael R. Garey and David S. Johnson. *Computers and intractability*. A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences. W. H. Freeman and Co., San Francisco, Calif., 1979, pp. x+338. ISBN: 0-7167-1045-5.

[4] Peter W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *35th Annual Symposium on Foundations of Computer Science (Santa Fe, NM, 1994)*. IEEE Comput. Soc. Press, Los Alamitos, CA, 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700. URL: https://doi.org/10.1109/SFCS.1994.365700.

[5] Petar Ivanov and José Felipe Voloch. "Breaking the Akiyama-Goto cryptosystem". In: *Arithmetic, geometry, cryptography and coding theory*. Vol. 487. Contemp. Math. Amer. Math. Soc., Providence, RI, 2009, pp. 113–118. DOI: 10.1090/conm/487/09528. URL: https://doi.org/10.1090/conm/487/09528.

[6] David S. Dummit and Richard M. Foote. *Abstract algebra*. Third. John Wiley & Sons, Inc., Hoboken, NJ, 2004, pp. xii+932. ISBN: 0-471-43334-9.

[7] William Stallings. *Cryptography and Network Security. Principles and Practices*. Fourth. Pearson/Prentice Hall, Upper Saddle River, NJ, 2006. ISBN: 0-131-87316-4.