BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering
and Communication

BACHELOR'S THESIS

Brno, 2021 Alexander Mogrovics

BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF TELECOMMUNICATIONS

ÚSTAV TELEKOMUNIKACÍ

WEB APPLICATION ON PAIRING-BASED CRYPTOGRAPHY

WEBOVÁ APLIKACE KRYPTOGRAFIE VYUŽÍVAJÍCÍ PÁROVÁNÍ

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR — **Alexander Mogrovics**
AUTOR PRÁCE

SUPERVISOR — **M.Sc. Sara Ricci, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Alexander Mogrovics ***ID:*** 204435
Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Webová aplikace kryptografie využívající párování

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte teorii eliptických křivek (tj. definici křivky, vlastnosti, operace nad eliptickou křivkou, bilineární párování a kryptografii využívající párování). Výstupem práce bude implementace webové aplikace umožňující spuštění operací bilineárního párování a základních kryptografických protokolů využívajících párování (např. 3-way Diffie-Hellmanův protokol). Jako výchozí bod lze využít programovací jazyk Sage. Webová aplikace může být postavena na předchozí studentské práci (desktopová aplikace pro kryptografii eliptických křivek). Součástí práce bude také manuál popisující funkcionalitu webové aplikace.

DOPORUČENÁ LITERATURA:

[1] Washington LC., “Elliptic curves: number theory and cryptography.” CRC press; 2008 Apr 3.

[2] Menezes AJ, Katz J, Van Oorschot PC, Vanstone SA. Handbook of applied cryptography. CRC press; 1996 Oct 16.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 31.5.2021

Vedoucí práce: M.Sc. Sara Ricci, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRACT

The aim of this thesis is to shed some light on elliptic curve cryptography via web application. It follows mathematical and algebraic principles of establishing elliptic curves over finite fields, performing computations over them and establishing bilinear pairings. It also describes the architecture and implementation of the web application, which uses website template with embedded SageMathCell in order to minimize required space. This application allows the user to preform computation of points on elliptic curve over finite field and their plot, plot of an elliptic curve over real numbers, operations over points of elliptic curves, to establish bilinear pairing, to establish 3-way Diffie–Hellman key exchange and to conduct a MOV attack.

KEYWORDS

Elliptic Curves, Bilinear pairings, Cryptography, SageMath, Web Application

ABSTRAKT

Cílem této práce je osvětlit kryptografii eliptických křivek za pomoci webové aplikace. Sleduje matematické a algebraické principy vytvoření eliptických křivek v množině konečných těles a následné počítání s nimi, včetně bilineárního párování. Práce také popisuje architekturu a impementaci webové aplikace, která je tvořena šablonou s vloženými SageMathCell prvky, aby byly minimalizovány nároky na úložiště. Tato aplikace umožňuje uživatelům provádět výpočet bodů eliptické křivky v množině konečných těles a vykreslení jejich grafu, vykreslení grafu v množině reálných čísel, provádění operací nad body eliptických křivek, vytvoření bilineárního páru, ustanovení výměny klíčů mezi třemi stranami pomocí Diffie–Hellman protokolu a realizaci MOV útoku.

KLÍČOVÁ SLOVA

Eliptické křivky, Bilineární párování, Kryptografie, SageMath, Webová aplikace

MOGROVICS, Alexander. *Web application on pairing-based cryptography*. Brno, 2021, 56 p. Bachelor's Thesis. Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Telecommunications. Advised by M.Sc. Sara Ricci, Ph.D.

ROZŠÍŘENÝ ABSTRAKT

Tato bakalářská práce se věnuje oblasti kryptografie eliptických křivek. Cílem bylo navrhnout a následně vytvořit webovou aplikaci, která by tuto oblast představila za pomoci základních operací nad body eliptické křivky, jako je sčítání bodů a skalární násobení, vykreslením grafu v oboru reálných čísel i konečných těles, ukázkou bilineárního párování (Weilovo a Tateovo párování), MOV útoku na diskrétní logaritmus a dvou kryptografických protokolů (distribuce klíčů pomocí ECDH s využitím párování a digitální podpis pomocí Boneh–Lynn–Shacham (BLS) protokolu).

Teoretická část práce si klade za cíl seznámit čtenáře s problematikou eliptických křivek jako takových. Obsahuje definici eliptických křivek a jejich reprezentaci pomocí funkčního předpisu. V oblasti kryptografie se eliptické křivky definují nad konečnými tělesy, která jsou taktéž představena. Dále jsou představeny operace na eliptických křivkách, a to sčítání bodů a skalární násobení. Sčítání bodů je popsáno grafickou formou včetně grafů ukazujících možné případy, které mohou nastat. Skalární násobení je pak de facto definováno jako specifický případ sčítání bodů, např. $3 \cdot P = P + P + P$. Důležitou částí práce je diskrétní logaritmus realizovaný eliptickými křivkami (Elliptic Curve Discrete Logarithm Problem, dále jen ECDLP). V rámci teoretické části práce je tento problém popsán ve své klasické formě, která je použita pro uvedení verze využívající eliptické křivky, tedy Elliptic Curve Diffie–Hellman (ECDH) protokolu. Následně je představeno bilineární párování. Bilineární párování musí splňovat jisté podmínky, které jsou v rámci této práce uvedeny. Typy bilineárního párování (symetrické × asymetrické, Weilovo × Tateovo) jsou rovněž uvedeny a popsány. Sekce bilineárního párování je s předchozími sekcemi spjata mimo jiné i definicí takzvaných pairing-friendly křivek, tedy křivek vhodných k realizaci bilineárního párování. Je představena výměna klíčů mezi třemi stranami pomocí Diffie–Hellman protokolu, podpisový protokol BLS a útok na diskrétní logaritmus MOV.

Kapitola SageMath je v kontextu práce součástí teoretické části. SageMath je klíčovým nástrojem implementace praktického řešení práce. Tato kapitola slouží představení tohoto programu. SageMath je možné užívat pomocí lokální instalace nebo pomocí online verze zvané SageMathCell, která zároveň umožňuje vkládání formuláře tohoto programu do vlastních webových stránek. Výpočty jsou pak provedeny na vzdáleném serveru, ne na lokálním stroji. SageMath obsahuje konstruktory pro sestavení eliptických křivek či konečných těles. Práci nejen se samotnou eliptickou křivkou, ale také jejími body je v rámci konzole programu velmi intuitivní, například pro součet bodů P, Q eliptické křivky stačí zadat `R = P + Q`. SageMath rovněž obsahuje metody pro vytvoření bilineárního párování.

Praktická část práce je věnována využití představených nástrojů k samotné implementaci. Jsou představeny navrhované postupy tvorby webové aplikace. Jedním

z možných postupů je využití lokální instalace aplikace SageMath a její spojení s vlastní webovou aplikací vytvořenou v Javě či C# pomocí Python skriptu. Toto řešení nebylo zvoleno z důvodu zvýšených nároků na velikost úložiště kvůli lokální istalaci SageMath. Implementované řešení využívá webovou stránku s vnořenými SageMathCell bloky, které jsou naformátovany pro uživatelský vstup a reprezentaci výsledných dat. Nevýhodou tohoto řešení je relativně malá úroveň úpravy vzhledu takto vloženého bloku, nicméně aplikace je i přes tuto překážku uživatelsky přívětivá. Hlavní stránka webové aplikace je složena z barevných dlaždic, přičemž každá dlažice se chová jako odkaz na samostatnou stránku obsahující konkrétní funkcionalitu. Implementované funkcionality jsou:

- Points on EC
- Plot of EC in $\mathbb{R}$
- Point addition
- Scalar multiplication
- Bilinear pairing
- 3-way Diffie–Hellman Exchange
- MOV attack
- BLS scheme

Ke konkrétním stránkám lze přistupovat i skrz drop-down menu v pravém horním rohu webové aplikace. Sekce **Points on EC** umožňuje konstrukci eliptické křivky definované nad konečným tělesem a zobrazuje rovnici eliptické křivky, zadané konečné těleso, řád křivky a seznam bodů křivky, a to včetně bodu v nekonečnu. Součástí této sekce je taktéž grafická reprezentace bodů v oboru konečného tělesa. Sekce **Plot of EC in** $\mathbb{R}$ umožňuje konstrukci eliptické křivky v oboru reálných čísel a její reprezentaci v kartouzském souřadnicovém systému. Sekce **Point addition** umožňuje konstrukci eliptické křivky definované nad konečným tělesem a obsahuje dva drop-down selektory pro výběr bodů zadané křivky. Po zadání těchto parametrů aplikace poskytuje součet dvou zadaných bodů. Sekce **Scalar multiplication** umožňuje konstrukci eliptické křivky definované nad konečným tělesem, drop-down selektor pro výběr jednoho bodu této křivky a textové pole určené k zadání celočíselného násobitele tohoto bodu. Následně aplikace poskytne výsledek skalárního násobení zadaného bodu a násobitele. Sekce **Bilinear pairing** umožňuje výběr z předpřipravených eliptických křivek definovaných nad konečnými tělesy. Uživatel následně následně zvolí bod křivky v $\mathbb{F}_p$ a bod v $\mathbb{F}_{p^k}$. Volitelné parametry a, b jsou celá čísla, sloužící jako násobitelé zadaných bodů. Zobrazeným výsledkem je pak hodnota vypočteného párování. Tato sekce obsahuje dva formuláře, jeden pro Weilovo párování a druhý pro Tateovo párování. Sekce **3-way Diffie–Hellman Exchange** umožňuje výběr z předpřipravených eliptických křivek definovaných nad konečnými tělesy, výběr řádu použitých bodů eliptické křivky, výběr samotných

bodů a zadání tajných klíčů a, b, c. Uživateli jsou pak poskytnuty informace o parametrech všech tří účastníků, tedy Alice, Boba a Charlieho. Těmito parametry jsou tajný klíč a veřejný klíč. Následně je zobrazena hodnota k, tedy výsledek bilineárního párování sdílených veřejných klíčů. Sekce **MOV attack** umožňuje výběr z předpřipravených eliptických křivek definovaných nad konečnými tělesy, výběr bodu zadané křivky P a výběr jeho násobku $a \cdot P$. Cílem této funkcionality je zjistit tajný klíč a. Další bod potřebný pro párování, $Q \in E(\mathbb{F}_{p^k})$, je vybrán náhodně. Následně je spočítána hodnota $e_1(P, Q)$ a $e_2(a \cdot P, Q) = e_2(P, Q)^a = {e_1}^a$ a je zobrazen výsledek a. Tento výsledek je možné ověřit ve formuláři pro skalární násobení, je potřeba zadat bod P a spočtené a. Výsledek by se měl shodovat s hodnotou $a \cdot P$. Sekce **BLS scheme** je realizována pomocí křivky BN254. Součástí je výpis generovaných parametrů, podpis a jeho ověření. Výsledek ověřovací fáze lze ovlivnit použitím nesprávného podpisu pomocí tlačítka Fake signature a nebo změnou zprávy.

DECLARATION

I declare that I have written the Bachelor's Thesis titled "Web application on pairing-based cryptography" independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the thesis and listed in the comprehensive bibliography at the end of the thesis.

As the author I furthermore declare that, with respect to the creation of this Bachelor's Thesis, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll., Section 2, Head VI, Part 4.

Brno ....

author's signature

ACKNOWLEDGEMENT

My deepest gratitude goes to my advisor M.Sc. Sara Ricci, Ph.D. for her guidance, patience, knowledge and motivation, especially during a global pandemic. Her input was always greatly appreciated and helpful, given her deep understanding of the topic and she made sure to thoroughly address whichever questions and doubts I might have had regarding the thesis and my ability to complete it, even if I thought I would not be able to go on anymore. I would like to thank my parents, my sisters Eva and Lucie, my friends and my dear partner Silvie for supporting and reassuring me. Their input made me work harder than I ever could without them.

Contents

List of Figures

Introduction

Ensuring security is a concept that has been present with humanity since the dawn of time. As we progressed as a society, the need to conceal certain information became crucial. With the rise of a modern society, our privacy came in danger. Now, every electronic device which connects to the Internet uses cryptography to a certain extent. This is the result of the very essence of the Internet as an open platform anyone can use.

Securing ones access to the Internet or securing communication and data can be performed in a number of ways. Our inspiration comes from history, when ciphers using the same key for encryption and decryption were used. Such cryptosystems are called symmetric ciphers. The task at hand is securing the key distribution as well. This is done by using asymmetric ciphers, which use a key pair, i.e. public and private key.

Asymmetric cryptosystem are implemented with certain key sizes. These key sizes are an indicator of the security of such system. As technological progress moves forward constantly, the keys need to be larger and larger. This is due to the fact, that asymmetric cryptography relies on NP problems such as integer factorization or discrete logarithm problem. While not impossible to compute, there is no known algorithm to solve them and immense computational power is needed to break keys. Elliptic curve cryptography provides a way of aiding this issue. For example, a 168-bit elliptic curve key is equivalent to a 2048-bit RSA key. However, computations on elliptic curves are not as straight-forward as is the case with standard asymmetric systems.

The goal of this thesis is to create a web-based application showcasing the use of elliptic curves in modern day cryptography. Web applications have become a widely used tool during the course of the last two decades. This is due to their instant accessibility on the Internet without the need to actually install the program locally and having to perform calculations on users own machine. Remote computing largely widens the scope of possible application features and computation complexity depending on the hosting system. In the context of this thesis, a simple HyperText Markup Language (HTML) website and SageMath program are used. SageMath, an open-source mathematics software, is however installed locally and would therefore require to build an infrastructure to connect it with the web application in order to perform calculations. A more straight-forward tool to overcome this is SageMathCell, which allows developers to embed Sage instructions directly into a website. The calculations are then performed on a dedicated Sage server. There is also the possibility of running your own server. SageMath includes a variety of tools for elliptic curve calculations, such as basic operations on the curve, calcula-

tion and listing of points, plotting of a graph, pairings and more. These tools are crucial, as the web application's intended features include showcasing of basic operations (point addition and scalar multiplication), plotting, operations necessary for bilinear pairings, 3-way Diffie–Hellman protocol and Boneh–Lynn–Shacham (BLS) Short Signature scheme.

In the **Background** chapter, the mathematical background of elliptic curves is introduced. Elliptic curves can be described by a number of forms. Important part of elliptic curve cryptography (ECC) are finite fields, over which the curves are defined. Basic operations on the curve are covered. They include point addition and scalar multiplication. Point addition is shown geometrically. Because ECC is based on computations with points on a curve, the order of the curve is covered as well. Discrete logarithm problem is introduced, represented by Diffie–Hellman key exchange scheme.

In the **SageMath** chapter, SageMath computational software is introduced. The main focus of this chapter is to show examples of elliptic curve constructors, algebraic operations on elliptic curves over finite fields and bilinear pairings. Web-based solution named SageMathCell is discussed, including the creation of individual interactive cells and their embedding into a website.

In the **Implementation** chapter, the proposed solution and used technology is described. Included are examples of used code and screen-shots of the created application. The proposed solution uses SageMathCell. Each feature of the application is described, including Sage methods used to create them.

1 Background

This chapter is meant to introduce the mathematical background of elliptic curves. At first, we shall look into defining elliptic curves as well as their definition over a finite field. This includes forms which can be used to represent the curve; primarily the **generalized Weierstrass form** and the **short Weierstrass form**. The main focus, however, will be put on describing basic operations on the curve (**point addition** and **scalar point multiplication**) with special attention to **bilinear pairings**.

Another focus of this chapter is the introduction of SageMath tool; its specifics, functions and possible uses.

The easiest and most user-friendly means of describing the curves and operations on them is doing so graphically. The next step would be algebraic definition. These definitions will help us with presenting the main focus of this chapter, bilinear pairings. Then Elliptic Curve Discrete Logarithm Problem (ECDLP) shall be introduced. From ECDLP we will move on to present cryptographic protocols, which utilize this problem and bilinear pairing: **Boneh–Lynn–Shacham Short Signature scheme** and **3-way Diffie–Hellman key exchange**.

1.1 Elliptic Curves

Elliptic curves can be written in several forms. Depending of the form, their properties and speed change. An elliptic curve $E(\mathbb{K})$ can be described by the following equation, which is called the **general Weierstrass form** of the elliptic curve [1]:

$$y^2 + \mathrm{a}_1 xy + \mathrm{a}_3 y = x^3 + \mathrm{a}_2 x^2 + \mathrm{a}_4 x + \mathrm{a}_6, \tag{1.1}$$

where $\mathrm{a}_1, \mathrm{a}_2, \mathrm{a}_3, \mathrm{a}_4, \mathrm{a}_6 \in \mathbb{F}_q$ are constant coefficients of the curve in a field $\mathbb{K}$. Elliptic curve is a set of points $(x, y) \in \mathbb{F}_q^2$ and a point at infinity noted as O that satisfies Equation 1.1. Elliptic curves can be constructed over the following fields $\mathbb{K}$:

$\mathbb{R}$ – real numbers,
$\mathbb{C}$ – complex numbers,
$\mathbb{Q}$ – rational numbers,
$\mathbb{F}_q$ – a finite field

Even though the general Weierstrass form allows to identify an elliptic curve, it is not commonly used in practice due to its speed [2]. Elliptic curves can also be described by a so called **short Weierstrass form**:

$$y^2 = x^3 + \mathrm{a}x + \mathrm{b}, \tag{1.2}$$

where a, b are constants in a field $\mathbb{K}$, where $a, b \neq 2, 3$. This form is commonly used in cryptography.

So far, we have established the curve as a set of points that satisfy the equation, along with a point at infinity. Using the short Weierstrass form, we can then define elliptic curves with the following definition:

$$E(\mathbb{K}) = \{(O)\} \cup \{(x, y) \in \mathbb{K} \times \mathbb{K} \mid y^2 = x^3 + \mathrm{a}x + \mathrm{b}\}, \tag{1.3}$$

Another form of elliptic curves, used especially with bilinear pairings (see Section 1.4), is the **Barreto–Naehrig form**.

$$y^2 = x^3 + b, \tag{1.4}$$

where $b \neq 0$ over $\mathbb{F}_p$, where p is a prime, with prime order and embedding degree $k = 12$ (for embedding degree, see Section 1.4.2).

1.1.1 Finite Field

In this section, we introduce the finite field, because that is what EC are defined over in cryptography. Finite field $\mathbb{F}$ is a set of elements and two binary operations. These operations are point addition $+$ and scalar multiplication $*$. Size of the set (in other words, number of elements) is called order of the finite field.

Definition 1.1.1. Let $\mathbb{F}$ be a set of elements on which two binary operations ($+$ and $*$) are defined. The set $(\mathbb{F}, +, *)$ is a **field** if the following conditions are satisfied:

- $(\mathbb{F},+)$ is a commutative group (and 0 is the identity element).
- $(\mathbb{F} \setminus \{0\}, *)$ is a commutative group (and 1 is the identity element).
- Multiplication is distributive over addition, i.e. for any three elements a, b and $c \in \mathbb{F}$,

$$a * (b + c) = a * b + a * c. \tag{1.5}$$

In cryptography, finite field are usually represented by $\mathbb{F}_q$ with $q = p^k$, where p is prime and $k \in \mathbb{N}$.

1.2 Algebra of Elliptic Curves

Algebra of elliptic curves defines two operations. They are point addition and scalar multiplication. These operations are what enables us to actually use ECC. All computations on the curves are in modulus p, which is a characteristic of $\mathbb{F}_p$, with desired high points of order.

1.2.1 Point addition

Given $P = (x_p, y_p)$ and $Q = (x_q, y_q)$ points on an EC $E(\mathbb{F}_p)$, the point addition [3, 4] $R = P + Q$ can be computed using the following procedure:

1. Draw a line through P and Q. The intersection of the line and curve E gives us point R'.
2. Reflect R' across the x-axis. This produces the desired point R.

Figure 1.1 shows $P + Q$ in case that $P \neq Q$.

Case of point addition where $P = Q$ would be **doubling**. Given $P = (x_p, y_p)$, doubling can be computed as $R = P + P = 2P$:

1. Draw a tangent line to E at $P = (x_p, y_p)$. Intersection of this tangent and curve E produces point R'
2. Reflect R' across the x-axis. This produces the desired point R.

Figure 1.2 shows $P + Q$ in case that $P = Q$. This operation can be performed multiple times and in such case is referred to as **successive doubling**. The addition of points on an elliptic curve E satisfies the following properties:

- Commutativity: $P_1 + P_2 = P_2 + P_1$ for all $P_1, P_2 \in E$.
- Existence of identity: $P + \infty = P$ for all $P \in E$.
- Existence of inverses: Given $P \in E$, there exists $P' \in E$, where $P + P' = \infty$. P' is normally denoted as $-P$.

The points on E are an additive group.

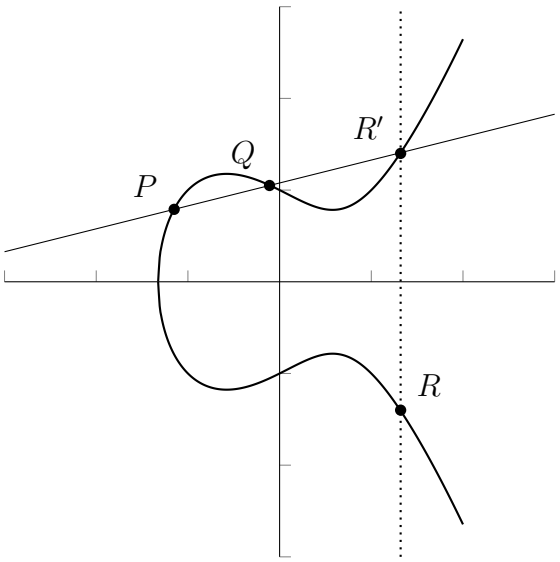

Fig. 1.1: Point addition where $P \neq Q$

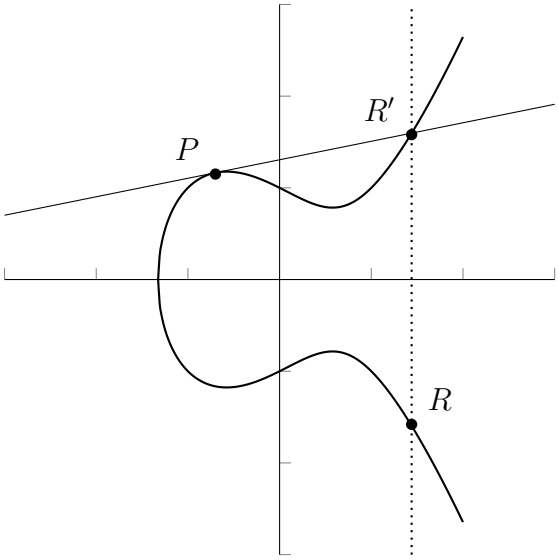

Fig. 1.2: Point doubling where $P = Q$

In case of point addition where $P = (x_p, 0) = Q$, the tangent to E in P is parallel to the y-axis. The resulting point is at infinity, as shown in Figure 1.3.

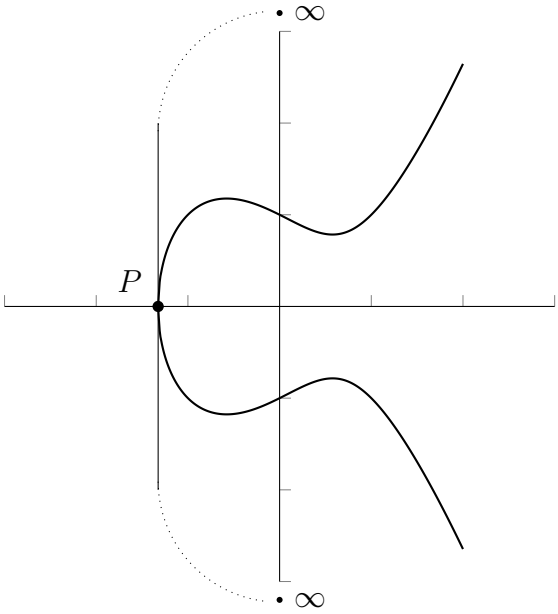

Fig. 1.3: Point addition where $P = (x_p, 0)$

Another case of point addition is for points $P = (x_p, y_p)$ and $Q = -P = (x_p, -y_p)$, as seen in Figure 1.4.

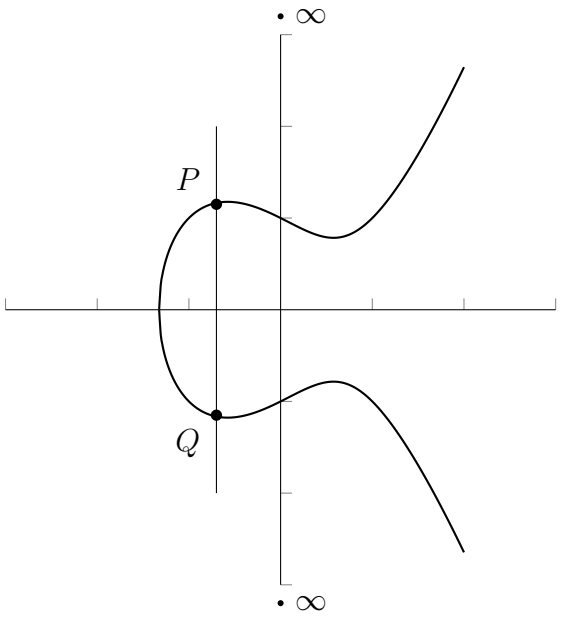

Fig. 1.4: Point addition where $P = (x_p, y_p)$ and $Q = (x_p, -y_p)$

1.2.2 Scalar multiplication

Scalar multiplication [3, 4] is a key feature of elliptic curves algebra and is what ECC relies on. If $\mathbb{F}_q$ is a finite field, we can conclude that there are finitely many sets of points (x, y) and the group $E(\mathbb{F}_q)$ is therefore finite. The scalar multiple s of point P is defined as following:

$$T = s \cdot P = \underbrace{P + P + \cdots + P}_{\text{s times}}. \tag{1.6}$$

If $P = O$, then $T = s \cdot O = O$ where O is a point at infinity. We can also use successive doubling, as shown in Section 1.2.1) for calculations using the scalar multiple of higher orders, for example $2P = P + P$, $3P = 2P + P$ and so on.
Example: Let s be 19 and P a point of EC. $T = s \cdot P = 19P$.

- $P + P + \cdots + P$ would require a total of 19 additions.
- $P + P = 2P$, $2P + 2P = 4P$, $4P + 4P = 8P$ and $8P + 8P = 16P$. Then we can compute $19P = 16P + 2P + P$ using only 6 additions.

1.2.3 Order of EC

If $\mathbb{K}$ is a finite field $\mathbb{F}_p$, where p is a modulus, there exists only a finite amount of points (including point at infinity $O = \infty$) and the curve E is therefore finite, as shown in Section 1.2.2).

Definition 1.2.1. The order of $E(\mathbb{F}_p)$ is the total amount of points on E, including the point at infinity O.

Order of E is not the same as the modulus. Let curve $E(\mathbb{F}_5)$ be $y^2 = x^3 - x + 1$ where $p = 5$:

x	$x^3 - x + 1 \pmod 5$	y^2	y	points
0	1	1	± 1	(0,-1), (0,1)
1	1	1	± 1	(1,1), (1,4)
2	2	-	-	
3	0	0	± 1	(3,0)
4	1	1	± 1	(4,-1), (4,1)
∞			∞	O

Tab. 1.1: $E(\mathbb{F}_5)$

As shown in Table 1.1, the order of $E(\mathbb{F}_5)$ is 8 while the modulus is 5.

1.3 Elliptic Curve Discrete Logarithm Problem

Modern asymmetric cryptography relies primarily on **integer factorization**, for instance in Rivest, Shamir, Adleman cryptosystem (RSA) and **discrete logarithm problem** (DLP), for instance in Diffie-Hellman protocol. The reason why these problems work so well in cryptography is simply due to the extreme difficulty of calculating them. These problems are hard to compute, but efficiently solvable with the knowledge of the secret. Shor's algorithm allows to solve integer factorization, DLP an ECDLP using quantum computers [6]. The time complexity of ECDLP is exponential, whereas the standard DLP has a sub-exponential cost. ECC also allows for use of comparably smaller sizes than comparable algorithm not utilizing EC [7, 8].

DLP can be represented as following. Let the logarithm be

$$k = log_b\ a.$$

Solution to this logarithm would then be

$$b^k = a,$$

which can be easily found if b and k are known. This changes with the use of a modulus p:

$$b^k \equiv a \pmod p,$$

where p is a prime number. If k is large enough and we know the values of $a,\ b$ and p, then k cannot be efficiently found.

In the case of ECDLP, the problem lays in the computation of points on EC over finite fields. Let us take point $P \in E(\mathbb{F}_q)$ as an example. It would be difficult to find k which would fulfil equation $kP = O$, where O is a point at infinity. That is the computation over a single point. In case of two points $P,\ Q \in E(\mathbb{F}_q)$ it would be difficult to find k such that $kP = Q$.

Definition 1.3.1. Let E be an elliptic curve over a finite field $\mathbb{F}_q$. Suppose there exist two points $P,\ Q \in E(\mathbb{F}_q)$ such that $Q \in \langle P \rangle$. To find k such that $Q = kP$ is a hard problem.

1.3.1 Diffie–Hellman over E

Elliptic Curve Diffie–Hellman protocol (ECDH) is a variation of DH. ECDH does not use modular exponentiation, but EC and public-private key pair. The key operation of ECDH is **scalar multiplication**, as seen in Section 1.2.2.

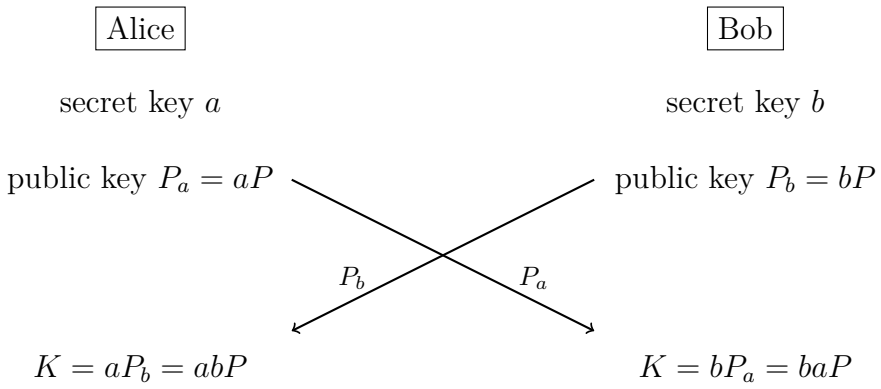

Fig. 1.5: Elliptic Curve Diffie–Hellman key exchange

Definition 1.3.2. Let E be an elliptic curve and P a generator of E of order n. Given P and aP it is hard to find a.

Decisional ECDH Assumption:

Let E be an elliptic curve and P a point of E. Given three points $Q, R, S \in E$, it is hard to decide whether there are integers n, m s.t. $Q = nP,\ R = mP$ and $S = nmP$.

Computational ECDH Assumption:

Let E be an elliptic curve and P a point of E. Given two points $Q = nP,\ R = mP \in E$, where n and m are unknown, it is hard to compute the value $S = nmP \in E$.

1.4 Bilinear Pairings

Bilinear pairings are an important part of ECC. It is used in elliptic curve based signature schemes, e.g. **Boneh–Lynn–Shacham Short Signature scheme**. Other uses of pairings include attacks on DLP (MOV attack), key agreement with multiple parties, key distribution or identity-based encryption.

Let n be a prime number. Let $G_1 = \langle P \rangle$ and $G_2 = \langle Q \rangle$ be additive groups of order n and G_T a multiplicative group of order n. A bilinear pairing is a non-degenerative map

$$e : G_1 \times G_2 \to G_T.$$

G_1, G_2 and G_T are usually cyclic groups. In the case of EC, G_1 and G_2 can be either EC or subsets of EC points. G_T is a finite field [9].

Definition 1.4.1. A bilinear pairing is a non-degenerative map $e : G_1 \times G_2 \to G_T$, which satisfies:

- Bilinearity

$$e(P_1 + P_2, Q) = e(P_1, Q) \cdot e(P_2, Q) \quad \text{for} \quad P_1, P_2 \in G_1,\ Q \in G_2$$
$$e(P, Q_1 + Q_2) = e(P, Q_1) \cdot e(P, Q_2) \quad \text{for} \quad P \in G_1,\ Q_1, Q_2 \in G_2$$

- Non-degeneracy

$$\text{for all } P \neq O : \exists Q \in G_2 \text{ such that } e(P, Q) \neq 1 \in G_T$$
$$\text{for all } Q \neq O : \exists P \in G_1 \text{ such that } e(P, Q) \neq 1 \in G_T$$

- Computability – the ability to algorithmically compute $e(P, Q)$ effectively for any $P, Q \in G$.

Mapping e therefore represents a relationship of two elements form one group to one element from a second group. If $G_1 = G_2 = G$ then the pairing is **symmetric** and map e is commutative for any $P, Q \in G$:

$$e(P, Q) = e(Q, P)$$

If $G_1 \neq G_2$, the pairing is **asymmetric**. Compliance with Definition 1.4.1 renders the mapping admissible.

Given $P, Q \in E(\mathbb{F}_q)$, where $a, b \in \mathbb{F}_q$, the main properties of bilinear pairings are:

- $e(P, O) = e(O, Q) = 1$
- $e(-P, Q) = e(P, -Q) = e(P, Q)^{-1}$
- $e(aP, Q) = e(P, Q)^a = e(P, aQ)$
- $e(aP, bQ) = e(P, Q)^{ab}$
- $e(P, Q_1 + Q_2) = e(P, Q_1) \cdot e(P, Q_2)$

1.4.1 Weil and Tate pairings

Weil pairing is such a pairing, that satisfies the properties above. Let E be an elliptic curve defined over a field $\mathbb{K}$ and n a positive integer. Then Weil pairing is

$$e_W : E[n] \times E[n] \to \mu_n,$$

where $\mu_n = \{x \in \overline{\mathbb{K}} \mid x^n = 1\}$ is a cyclic group of order n.

Weil pairing needs the full n-torsion on order to work: $E[n] \subseteq E(\mathbb{F}_q)$, which implies that $\mu \subseteq \mathbb{F}_q$. Let $P \in E(\mathbb{F}_q)[n]$, $Q \in E(\mathbb{F}_q)$ and $R \in E(\overline{\mathbb{F}_q})$, satisfying $n \cdot R = Q$. ϕ_q is the q-th power Frobenius endomorphism. Then **Tate pairing** [1] is defined as

$$e_T(P, Q) = e_W(P, R - \phi_q(R)),$$
$$e_T : E(\mathbb{F}_q)[n] \times E(\mathbb{F}_q)/nE(\mathbb{F}_q) \to \mu_n.$$

1.4.2 Embedding Degree

From the perspective of cryptography, embedding degree is what tells us whether the curve provides strong security or not. In other words, it represents how difficult it would be to turn ECDLP into DLP. However, due to limiting factors such as technology and time efficiency, the embedding degree must be regulated. If it was unreasonably high, the computation of discrete logarithm represented by $E(\mathbb{F}_p)$ would be infeasible, but at the same time it would be more secure [10].

Definition 1.4.2. Let k be the embedding degree of $E(\mathbb{F}_p)$ and let n be a large prime dividing the order of $E(\mathbb{F}_p)$. Then

- k is the smallest integer s.t. $n|(p^k - 1)$,
- k is large enough to make computation of discrete logarithm in $\mathbb{F}_p$ infeasible. However, k is small enough to make the pairing easy to compute.

Different representation of $n|(p^k - 1)$ is $p^k \equiv 1 \bmod n$.

Example:
Let $E(\mathbb{F}_5) : y^2 = x^3 - x + 1$ be an elliptic curve with order 8.

$E(\mathbb{F}_5) : y^2 = x^3 - x + 1$

Points	Order
O	1
(3,0)	2
(4,4), (4,1)	4
(0,1), (0,4), (1,1), (1,4)	8

We have defined that the embedding degree k is the smallest integer such that $n|(p^k - 1)$. $\mathbb{F}_5$ gives the value of $p = 5$.

For $n = 4$ and $p = 5$:

$$n|(p^k - 1) = 4|(5^k - 1) \ \rightarrow \ 5^k \equiv 1 \bmod 4,$$

therefore $k = 1$.

For $n = 8$ and $p = 5$:

$$8|(5^k - 1) \ \rightarrow \ 5^k \equiv 1 \bmod 8,$$

therefore $k = 2$.

1.4.3 Pairing-friendly Elliptic Curves

Three common types of pairing-friendly curves are

- Supersingular curves (embedding degree $k \leq 6$),
- Miyaji–Nakabayashi (MNT) curves of prime order and small embedding degrees of 3, 4 and 6,
- Barreto–Naehrig (BN) curves (Equation 1.4, embedding degree $k = 12$).

Definition 1.4.3. Let $E(\mathbb{F}_q)$ be an elliptic curve, where $q = p^h$ and p is a prime. Let $a = q + 1 - \text{ord}\,[E(\mathbb{F}_q)]$. Then E is supersingular if and only if $a \equiv 0 \bmod p$, which is of and only if $\text{ord}\,[E(\mathbb{F}_q)] \equiv 1 \bmod p$.

Example of a supersingular curve
Let $E(\mathbb{F}_p)$, where $p > 5$ and the order of $E(\mathbb{F}_p) = p + 1$ be supersingular.

$$E(\mathbb{F}_p): \ y^2 = x^3 + 1,$$

where $p \equiv 2 \bmod 3$. Supersingular curves with $p < 5$ exist as well, e.g. $E(\mathbb{F}_2)$ and $E(\mathbb{F}_3)$.

$$E(\mathbb{F}_2): \ y^2 + y = x^3 + x,$$
$$E(\mathbb{F}_3): \ y^2 = x^3 - x + 2.$$

1.4.4 3-way Diffie–Hellman protocol

3-way Diffie–Hellman protocol [11, 12] works similarly to standard ECDH, but in this case with the use of bilinear pairings. Every participant of the key exchange communication transmits only one broadcast message.

1. Participants agree on a pairing $e : E[n] \times E[n] \rightarrow \mathbb{F}_n$ and $P, Q \in E[n]$, where $n = p^k$.

2. Alice chooses **private key** $a \in \mathbb{F}_n$ and broadcasts her **public key** $A = a \cdot P$.
3. Bob chooses **private key** $b \in \mathbb{F}_n$ and broadcasts his **public key** $B = b \cdot P$.
4. Charlie chooses **private key** $c \in \mathbb{F}_n$ and broadcasts his **public key** $C = c \cdot P$.
5. Alice computes $k = e(bP, cP)^a = e(P, P)^{abc}$, Bob computes $k = e(aP, cP)^b = e(P, P)^{abc}$ and Charlie computes $k = e(aP, bP)^c = e(P, P)^{abc}$.

k is the shared secret key all three parties have computed.

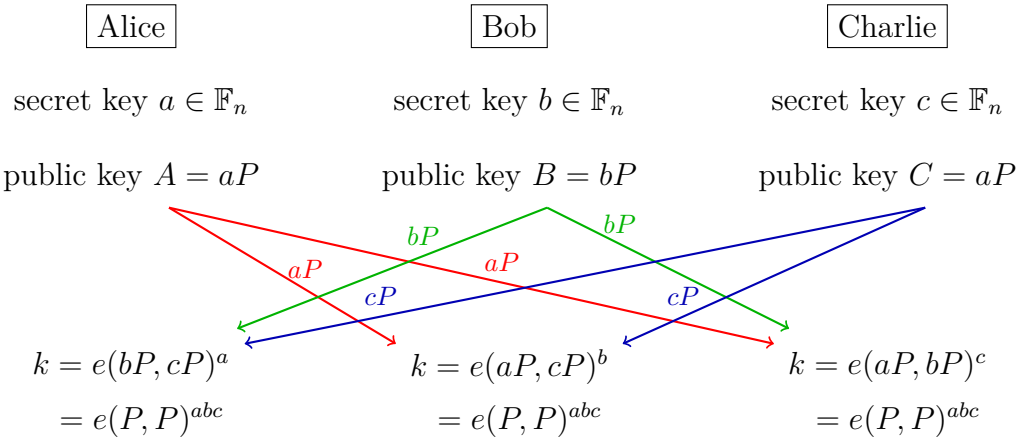

Fig. 1.6: 3-way Diffie–Hellman key exchange

1.4.5 Boneh–Lynn–Shacham Signature Scheme

Boneh–Lynn–Shacham (BLS) short signature scheme [13] was designed by Dan Boneh, Ben Lynn and Hovav Shacham as a means of providing a short enough signature with to use over low-bandwidth channels. Generation of the signature is done by relatively simple scalar multiplication of a point on an elliptic curve. Verification is done by comparison of bilinear pairings. This scheme was designed with Weil pairing in mind.

In order to sign the message, a hashing function is needed. Because the signed message must be a point on EC, it is necessary to map the hash to the proper field.

$$h : \{0, 1\}* \to \mathbb{F}_{p^k}$$

For each valid x coordinate, there are two possible points. One with positive y coordinate and the other with negative y. In order to find a valid point, a value is appended to the message: $h(m||i)$, where h is the hashing fuction, m is the message and i is the incremental value. If no point is found, the value is incremented. Probability of the hash being an affine point is approximately $\frac{1}{2}$.

1. **Key generation**: Let $P \in E(\mathbb{F}_{p^k})$ be a point of order q and random $x \in \mathbb{Z}_q$. The public key is $PK = x \cdot P$ and x is the private key.

2. **Signature**: Let m be a message. Hash m to the curve using $H = h(m)$. The signature is $S = x \cdot H$.
3. **Verification**: Let e_1, e_2 be bilinear pairings. To verify the signature, compare $e_1(PK, H)$ and $e_2(P, S)$. Verification is successful only if $e_1 = e_2$, because of the following properties:

$$e_1(PK, H) = e_1(xP, H) = e_1(P, H)^x$$
$$e_2(P, S) = e_2(P, xH) = e_2(P, H)^x$$

1.4.6 MOV attack

Attack on the ECDLP is called MOV (Menezes, Okamoto and Vanstone) [14]. It's mechanism is to reduce the discrete logarithm problem to a easier discrete logarithm problem. MOV attack only works with symmetric pairing.

This is achieved by Weil pairing. Weil pairing converts the ECDLP in $E(\mathbb{F}_q)$ to DLP in $\mathbb{F}_{q^m}$.

Let $e: \ G \times G \to G_T$ be a pairing. Given $P \in G$ and $aP \in G$ it is possible to solve $e(P, P) \in G_T$ as well as $e(P, aP) = e(P, P)^a \in G_T$.

$e(P, P) = g$ and $e(P, P)^a = g^a$, where $g \in G_T$ and therefore this attack can be used in G_T to find a.

Example:

Let $P \in E(\mathbb{F}_{23})$ be a point of order 10 and

$$e: \ E[10] \times E[10] \to \mathbb{F}_{11}$$

a bilinear pairing. If $e(P, P) = 2$ and $e(P, Q) = 5$, then it is possible to find a:

- Compute the powers of generator $e(P, P) = g = 2$ in $\mathbb{F}_{11}$ until $g^a = 5$ is found.

$$2^1 \equiv 2 \bmod 11$$
$$2^2 \equiv 4 \bmod 11$$
$$2^3 \equiv 8 \bmod 11$$
$$2^4 \equiv 5 \bmod 11$$

- $e(P, aP) = e(P, P)^a = g^a = 2^4$ and $e(P, P) = g = 2$. Therefore $a = 4$.

2 SageMath

SageMath is a free open-source mathematics software. SageMath's computational abilities include a vast number of mathematical fields, such as basic algebra, calculus, number theory, cryptography, numerical computation, commutative algebra, group theory, combinatorics, graph theory and so on. It is available for Windows, Linux and MacOS. In essence, it functions as a free alternative to programs such as Maple or Matlab.

Sage also offers an online solution titled SageMathCell[1]. SageMathCell is an open-source web interface, that allows users to perform Sage computations without having to install Sage. It can also be easily embedded into any webpage [16].

Sage is capable of performig computations over EC. Built-in functionalities include defining a curve (including the field), calculating the order of the curve, points on the curve and their orders as well as point addition and scalar multiplication.

2.1 Elliptic Curves in Sage

There are multiple ways of defining an elliptic curve in Sage. The command for defining EC is `EllipticCurve`, which can have a plethora of parameters [15].

By default, Sage uses the generalized Weierstrass form described in Equation 1.1. What defines the curve are the parameters a_1, a_2, a_3, a_4 and a_6. In Sage they are entered in the same order as in Equation 1.1.

```
#command defining an EC
EllipticCurve([a1, a3, a2, a4, a6])
```

The previous command would define the curve over $\mathbb{R}$ [17]. To define a curve $E(\mathbb{F}_5): \ y^2 = x^3 - x + 1$, one would need to input

```
E = EllipticCurve(GF(5),[0, 0, 0, -1, 1])
```

,where `GF(5)` defines $\mathbb{F}_5$.

Sage has an built-in tool, which returns the order of an elliptic curve. The function is titled `cardinality()`. The following syntax is used:

```
#definition of EC
E = EllipticCurve(GF(5),[0, 0, 0, -1, 1])

#calling cardinality() over E
E.cardinality()
```

[1] `https://sagecell.sagemath.org/`

2.2 Points and operations

Points on an elliptic curve can also be returned by their own function, `points()`.

```
#definition of EC
E = EllipticCurve(GF(5),[0, 0, 0, -1, 1])

#calling points() function
E.points()
```

This function returns an array of points on a selected elliptic curve. Sage displays points on an elliptic curve in two formats.

- $(x : y : 0)$ is a point at infinity O.
- $(x : y : 1)$ is an projective point.

These formats are not suitable as an output. In order to get the points in (x, y) form, a method `.xy()` can be called over a point. This method cannot be used over a point at infinity

Example:

```
Input:
E = EllipticCurve(GF(5),[0, 0, 0, -1, 1])
print(E)
print('Order of E: ', E.cardinality())
print('Points: ', E.points())
print('Formatted points:')
#iterate through all points
for point in E.points():
  #check for point at infinity (x : y : 0)
  if point[2] != 0:
    print(point.xy())

Output:
#elliptic curve
Elliptic Curve defined by y^2 = x^3  - x + 1 over Finite
Field of size 5
#cardinality
Order of E: 8
#array of points
Points:  [(0 : 1 : 0), (0 : 1 : 1), (0 : 4 : 1),
          (1 : 1 : 1), (1 : 4 : 1), (3 : 0 : 1),
          (4 : 1 : 1), (4 : 4 : 1)]
```

```
Points in affine coordinates:
(0, 1)
(0, 4)
(1, 1)
(1, 4)
(3, 0)
(4, 1)
(4, 4)
```

The defined curve has an order of 8 with 7 projective points and one point at infinity.

Point addition is also already implemented in Sage and is as simple as `P + Q`, after defining points P and Q. Sage then returns the resulting point on the curve. Example of point addition using the curve from the previous example:

```
Input:
P = E(0,4)
Q = E(1,4)
print('Point P = ', P.xy())
print('Point Q = ', Q.xy())
R = P + Q
print('Sum of P,Q: R = ', R.xy())

Output:
#point P
Point P = (0, 4)
#point Q
Point Q = (1, 4)
#P+Q
Sum of P,Q: R = (4 , 1)
```

Sage supports scalar multiplication as well. Requirements for scalar multiplication are a curve E, point on the curve P and scalar multiple s.
Example of scalar multiplication using the curve and point P from the previous example:

```
Input:
print('Point P = ', P.xy())
s = 4
print('Scalar multiple s = ', s)
T = s*P
print('Result of multiplication: T = ', T)
```

```
Output:
#point P
Point P = (0, 4)
#scalar multiple s
Scalar multiple s = 4
#s*P
Result of multiplication: T = (3, 0)
```

2.3 Bilinear Pairings

There are three types of pairings implemented in Sage. They are Ate, Tate and Weil pairings. For the purpose of this thesis, Tate and Weil are used. The following parameters are necessary to successfully compute the pairings:

- Scalar multiples[2] a, b.
- P, Q are points on curve $E(\mathbb{F}_q)$.
- r is an integer s.t. $n \cdot P = n \cdot Q = (\infty, \infty)$.
- n is the order of point P.
- k is the embedding degree.

```
#Weil pairing
e_weil = (a*P).weil_pairing(b*Q, r)

#Tate pairing
e_tate = (a*P).tate_pairing(b*Q, n, k)
```

2.4 SageMathCell

SageMathCell is an web-based version of SageMath. It does not require a local installation, because it is hosted and running on a remote server. Embedding SageMathCell is a fairly simple task. The following line needs to be added into the `<head>` section of an HTML file of the website:

```
<script src="https://<server>/static/embedded_sagecell.js"></script>
```

where the default `<server>` is `sagecell.sagemath.org`. It is possible to create a personal SageMathCell server and use it this way.

In order to create a cell, the following script should be run:

[2]These parameters are not mandatory.

```
sagecell.makeSagecell({inputLocation: "[jQuery selector]"});
```

where `[jQuery selector]` can be a predefined input location, such as `div.compute`, which would create a Sage cell into which the user input could be entered. It can also be user defined, such as `#mycell`. This would affect all cells created with `id=mycell`.

The language in which the computational commands are entered is Python. SageMathCell also supports interactive input. That is achieved by the `@interact` command. A number of different input types can be created, for example text boxes.

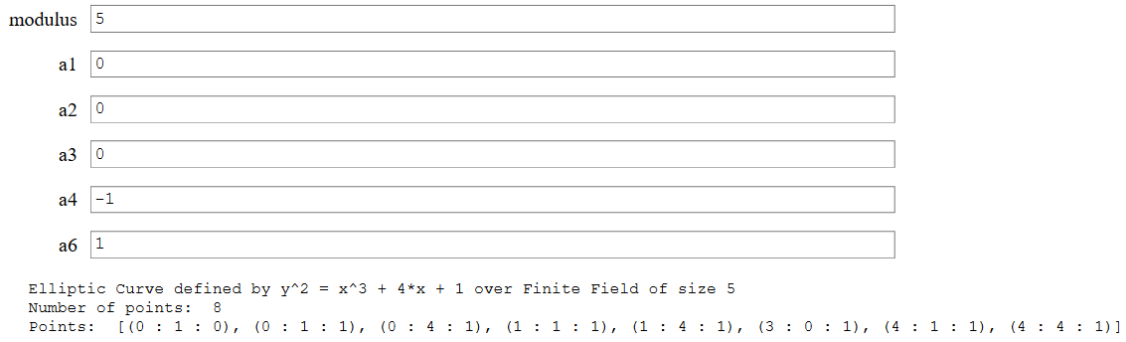

Fig. 2.1: Textboxes with default values

```
@interact          #default value p=5
def _showPoints(p = input_box('5', label='modulus'),
                   #default value a1=0
                a1 = input_box('0', label='a1'),
                a2 = input_box('0', label='a2'),
                a3 = input_box('0', label='a3'),
                a4 = input_box('-1', label='a4'),
                a6 = input_box('1', label='a6')):
    E = EllipticCurve(GF(p),[a1, a3, a2, a4, a6])
    print(E)
    print('Number of points: ', E.cardinality())
    print('Points: ', E.points())
```

This code would produce six text boxes with default values, as seen in 2.1.

Input types and selectors can be combined. For example, it is possible to create one text box selector and make the other one a drop down menu, as seen in 2.2.

Fig. 2.2: Textbox and drop down menu selector

The code used to achieve this:

```
#default point from points() array
def _(P = selector(listOfPoints, default=listOfPoints[2],
      label='Point P'),
      n = input_box('1', label='n')):
```

3 Implementation

Implementation of this thesis is focused on creating a web application, which would serve as a educational and showcasing tool about basic operations on elliptic curves, such as point addition and scalar multiplication, as well as pairing operations. Operations with pairings are best shown in the form of established protocols, e.g. Boneh–Lynn–Shacham Signature Scheme or ECDH.

Recommended computational program for this implementation was Sage, which only offers console input and output. That is not suitable for such application, let alone a web application as it is not user-friendly and would make the overall comfort of interacting with the application minimal. It is, however, suitable for background computations.

The biggest challenge lies in successfully connecting Sage to an existing website. Once that is achieved, the user would be able to use the website to:

- generate input in the form of variables
- have the input data calculated by Sage
- see the results of Sage calculations

However, certain limitations regarding connection appear with this solution as Sage needs to work as a back-end for the front-end (the website). Connecting the two would be possible with the use of a custom Python script, but there is also the option to use SageMathCell, which does not require a local installation as it is installed on a remote server.

The application consist of multiple tabs, which are described in next sections. List of all pages:

- Points on EC
- Plot of EC in $\mathbb{R}$
- Point addition
- Scalar multiplication
- Bilinear pairing
- 3-way Diffie–Hellman Exchange
- MOV attack
- BLS scheme

The solution is available at `https://www.stud.feec.vutbr.cz/~xmogro00/`.

3.1 Web Application

The goal of this thesis is to create a web application showing pairing based cryptography. In order to do that, it is neccessary to connent the web page with Sage.

While Sage is primarily a desktop application, SageMathCell can be embedded into web pages.

The initial solution was based on the one created in [18]. The author had to create a connection between JavaFX application and a desktop version of Sage. The author overcame this problem by using sockets. In that scenario, SageMath serves as a back-end and JavaFX application as a front-end. Because Sage is written in Python and is capable of running Python scripts, the author created a script, which set up a listener on SageMath Server and was taking requests from the JavaFX client.

For the purpose of this thesis, the initial solution tried to replicate that using Spring[1] framework in a Java application, which would send requests to a running instance of SageMath and display formatted results. This solution soon proved impractical, since the aim was to deploy the solution to a website where Sage is not installed and would require to install and run Sage on the same server, that is hosting the web application or to be constantly connected to a server running Sage. Therefore, SageMathCell was used.

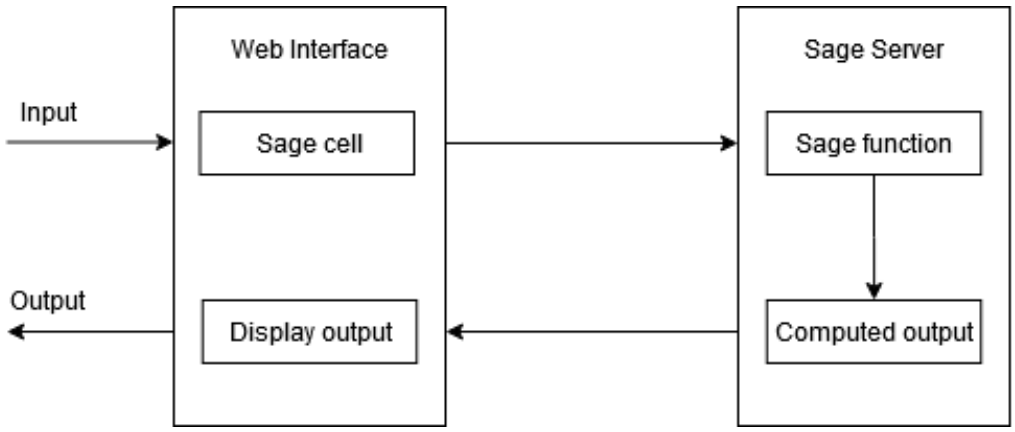

Fig. 3.1: Diagram of internal logic

Once SageMathCell is embedded into a website, user can generate input using the created cell on the website. This input sent via the Internet to a selected SageMathCell server, which performs the computations described in the created cell. These computations are written in Python. After the computations are performed, SageMathCell server sends the data back to the website, which displays the results.

[1] `https://spring.io/`

3.1.1 Design

The website is based on a template available at `https://html5up.net/phantom` under Creative Commons licence (CC BY 3.0). This template was chosen for its simplicity, which makes it ease to navigate and generally user-friendly. The application was developed using WebStorm 2020.2.3.

Main menu is composed of colourful square tiles. They act as buttons and provide a link to specific pages. The location (functionality) is written in their center. When a user hovers over the tiles, additional description can be seen. Figure 3.2 shows an example of the design and styling of the tiles. All tabs are shown in the Appendix A.

Fig. 3.2: Example of main menu tiles

Each separate page contains a drop down menu to help access to other pages. The user can therefore either use the „Go Back“ button in their browser, click the link in the header, which leads to main menu, or use the drop down menu.

Example of tile configuration in the CSS file:

```
.tiles article {
      transition: transform 0.5s ease, opacity 0.5s ease;
      position: relative;
      width: calc(33.33333% - 2.5em);
      margin: 2.5em 0 0 2.5em;
}
```

Example of menu implementation in HTML, which is present on all pages:

```
<!-- Menu -->
<nav id="menu">
  <h2>Menu</h2>
    <ul>
      <li><a href="points.html">Points on EC</a></li>
      <li><a href="plot.html">Plot</a></li>
      <li><a href="pointadd.html">Point Addition</a></li>
      <li><a href="scalmult.html">Scalar
                                  Multiplication</a></li>
      <li><a href="pairing.html">Bilinear
                                 pairing</a></li>
      <li><a href="dhexchange.html">DH Exchange</a></li>
      <li><a href="mov.html">MOV attack</a></li>
    </ul>
</nav>
}
```

3.2 Points on EC tab

This tab allows for user input of the modulus of a finite field and parameters of a curve. The output consists of EC equation, order of EC and an array of points on the curve including their orders. Also included is a plot of the curve in $\mathbb{F}_p$, see an example for $E(\mathbb{F}_5) : y^2 = x^3 - x + 1$ in Figure 3.3.

The implementation of this functionality utilizes only basic Sage commands and an iterator used for displaying points, as seen in Section 2.2.

3.3 Plot an EC tab

In this tab, the user can define their own EC over a rational field $\mathbb{R}$ an plot its graph. *Example:*

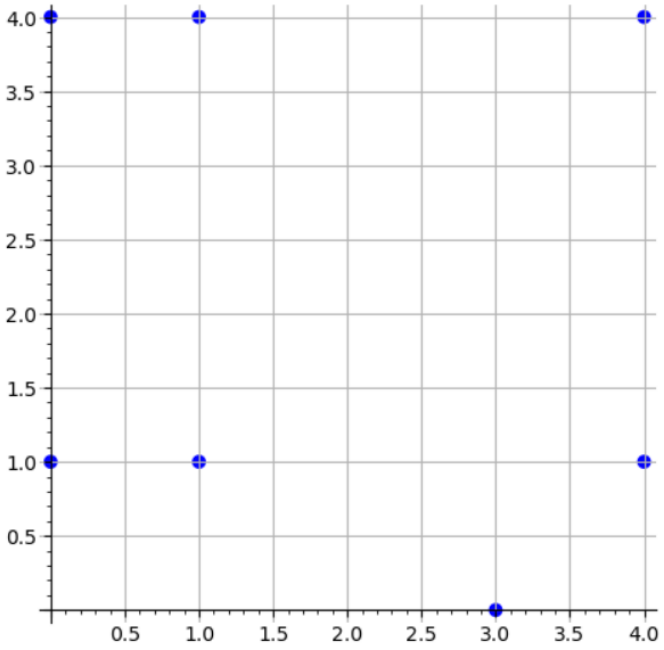

Fig. 3.3: Example of a plot in $\mathbb{F}_p$

```
show(graphics_array([plot(E,thickness=3)]),xmin=-3, xmax=3, ymin=-3,
 ymax=3,frame=True)
```

where E is a predifined elliptic curve, `thickess` is the thickness of the plotted graph and `xmin`, `xmax`, `ymin` and `ymax` are the graph borders. For the resulting plot, see Figure 3.4.

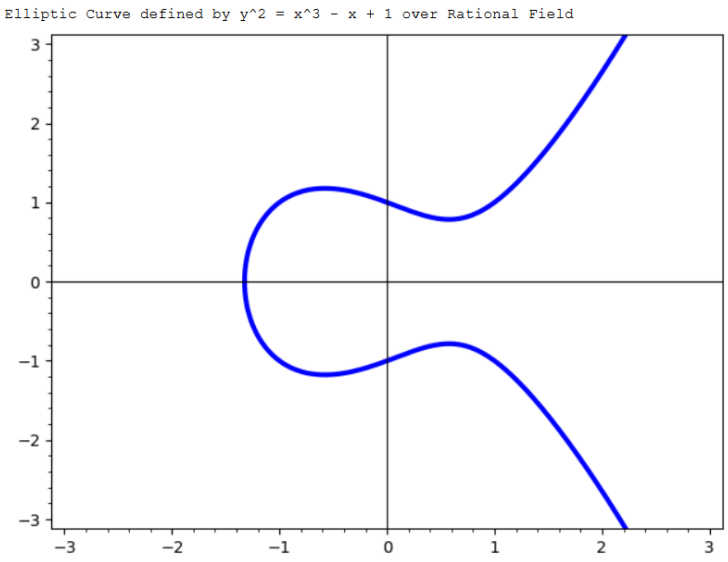

Fig. 3.4: Plotting in Sage

3.4 Point addition tab

Point addition tab presents the user with a predefined curve $y^2 = x^3 - x + 1$ over $\mathbb{F}_5$, but there are text boxes to define a different curve. User can choose points P and Q on the curve and their sum is calculated. SageCell uses the `@interact` keyword to determine if a method has interactive arguments (meaning it is waiting for user input). The curve needs to be selected first, then a list of points is calculated and only after that it can be passed as an argument to a method performing the addition. Figure 3.5 shows point addition on the created website.

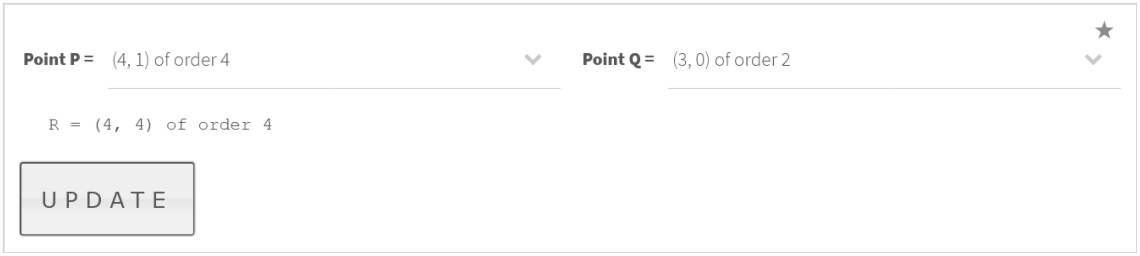

Fig. 3.5: Example of point addition

In order to be able to select the points, this method also needs the `@interact` handle, which creates a bordered window within the window of the previous handle. As a result, the final Sage form is composed of nested windows.

3.5 Scalar multiplication tab

This tab uses the predefined curve $y^2 = x^3 - x + 1$ over $\mathbb{F}_5$ as well. User chooses point P and a scalar multiple s. The default value of s is 1. Then, the multiplication is calculated and the result displayed. Window containing the output and point selection as again nested in the curve selection window. Figure 3.6 shows scalar multiplication on the created website. User selects two points from the drop-down menus and pushes the update button to perform the computation.

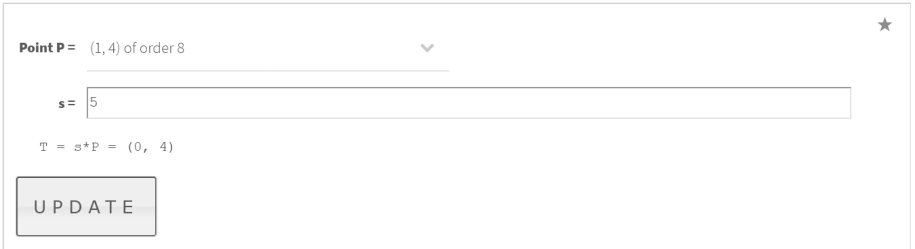

Fig. 3.6: Example of scalar multiplication

3.6 Bilinear pairing tab

As the name suggest, this tab allows the user to perform computations of bilinear pairing. Two pairing are used: **Weil pairing** and **Tate pairing**. User is able to select one curve form a list of three curves in the case of Weil pairing and two curves in the case a Tate pairing.

- $E(\mathbb{F}_{11}) : y^2 + y = x^3 + x + 1$ (only Weil pairing)
- $E(\mathbb{F}_{11}) : y^2 + 2y = x^3 + 3x + 3$
- $E(\mathbb{F}_{13}) : y^2 = x^3 + 36x + 5$

The user is provided with information regarding the selected curve, including a pre-selected torsion point, embedding degree and general formula of a pairing. Then users can select two points, that they want to use with pairing, and scalar multiples a, b. Figure 3.7 shows the Weil pairing section of the website.

Fig. 3.7: Weil pairing

Important property of Tate pairing is, that the two selected points must be of the same order. Therefore, users select a given curve, select desired order of points and only then can they select the points.

- `_setCurve(curveE)`. This method takes argument `curveE`, which is a selector and has arguments `selector(source, default_value, label)`.
  - `curveDict`, which is a dictionary containing elliptic curves.
  - `default=curveSelector[0]`, which sets the default value of the selector.

  - `label='EC: '`, which sets what is displayed in the form.
- `_orderSelect(order)`. This method takes argument `order`, which is also a selector. Only Tate pairing uses this method.
  - `orderDict`, which is a dictionary containing all orders of points.
  - `default=orderSelector[1]`, which sets the default value of the selector.
  - `label='Order of points = '`, which sets what is displayed in the form.
- `_pairing(pointP, pointQ, a, b)`. Arguments `pointP` and `pointQ` are again selectors, this time they contain values from dictionaries filtered in method `_orderSelect`. `a` and `b` are input boxes and contain integers.

3.7 3-way Diffie–Hellman Exchange tab

Implementation of 3-way DH protocol is conceptually similar to the implementation of the Tate pairing. User selects a pre-defined curve from a list. Then, they need to select a desired order of points on the curve. The secret keys a, b and c can be typed into accordingly labelled text boxes.

This function uses methods `_setCurve(curveE)` and `_orderSelect(order)`, as seen in Section 3.6. A new method `_exchange(a, b, c, pointP, pointQ)` has arguments `a`, `b`, `c`, which are selectors of values of $\mathbb{F}_p$ and selectors of points P and Q.

As output, the user sees parameters of all three participants and computed k. Figure 3.8 shows parameter selection. The scheme starts once the user presses the update button.

Fig. 3.8: Parameter selection

Example of and exchange with EC $E(\mathbb{F}_{11}) : y^2 + 2y = x^3 + 3x + 3$, $P = (t^2 + 8t + 10, 7t^2 + 3t + 1)$ and $Q = (9, 9)$, where P, Q are of order 7 with parameter selection shown in Figure 3.8:

```
Alice:
 SECRET key a = 6,
 PUBLIC key pair Ap = a * P = (y^2 + 8*y + 10, 4*y^2 + 8*y + 8)
                 Aq = a * Q = (4, 4)
Bob:
 SECRET key b = 2,
 PUBLIC key pair Bp= b * P = (7*y^2 + 9*y + 8, 10*y^2 + 6*y + 5)
                 Bq = b * Q = (8, 9)
Charlie:
 SECRET key c = 9,
 PUBLIC key pair Cp = c * P = (7*y^2 + 9*y + 8, 10*y^2 + 6*y + 5)
                 Cq = c * Q = (7, 6)

Alice broadcasts pair A = (Ap, Aq), Bob broadcasts B = (Bp, Bq)
and Charlie broadcasts C = (Cp, Cq).

Alice: k = 10*y^2 + 7*y + 8
Bob: k = 10*y^2 + 7*y + 8
Charlie: k = 10*y^2 + 7*y + 8
```

3.8 MOV attack tab

This tab provides basic information about the preliminaries of the MOV attack, such as two bilinear pairings and nature of necessary parameters. User is prompted to select an EC from a list, select a point P of the selected EC and select point $a \cdot P$.

Output consists of establishing pairings $e_1(P, Q)$ and $e_2(a \cdot P, Q) = e_2(P, Q)^a = e_1^a$, equation of the elliptic curve and embedding degree k. Point Q is randomly selected from $\mathbb{F}_{p^k}$. $\mathbb{F}_{p^k}$ is represented as $\mathbb{F}_p/(t^k + 1)$. Finally, the result a is displayed and user can check whether it is correct in a Sage cell right bellow by selecting the original point P and typing in the resulting a. Figure 3.9 shows parameter selection and the resulting output.

This tab uses methods similar to methods in other pairing related tabs, such as curve selection, point selection and computational method.

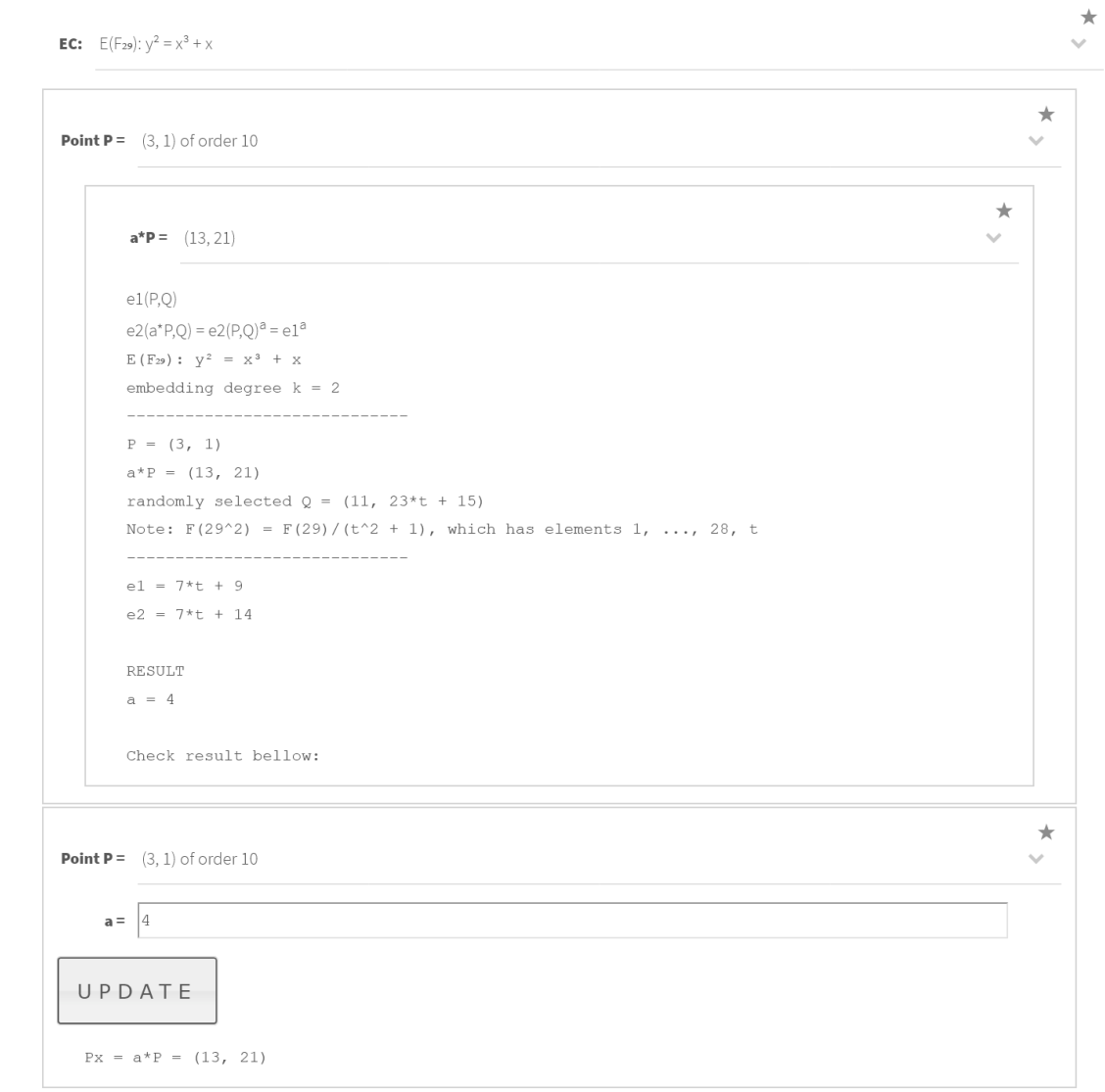

Fig. 3.9: MOV attack

3.9 BLS signature scheme tab

This tab shows the necessary steps in creating the BLS scheme. The scheme is implemented with a BN254 elliptic curve. User can enter an arbitrary message, which is then used in the computations.

Parameter generation shows randomly selected point P, private key sk and public key $PK = sk \cdot P$. A signature is generated by hashing of the message to the curve and multiplying it by the private key: $S = sk \cdot H(m)$, where S is the signature, m is the message and hashing function $H(m)$. Then, the signature is verified using bilinear pairings.

Users can alter the result in two ways. It is possible to tick a check-box labelled „Fake signature“, which uses altered (and incorrect) signature. There is also an input box, where the message m can be changed, as seen in 3.10.

Main method of this tab is `_setCurve(message)`, where the computations are performed and which includes methods `_sign(checkbox)` and `_verify(message)`. These auxiliary methods are mostly used to generate text with the computed values, but pairings are both computed and compared within the verification method.

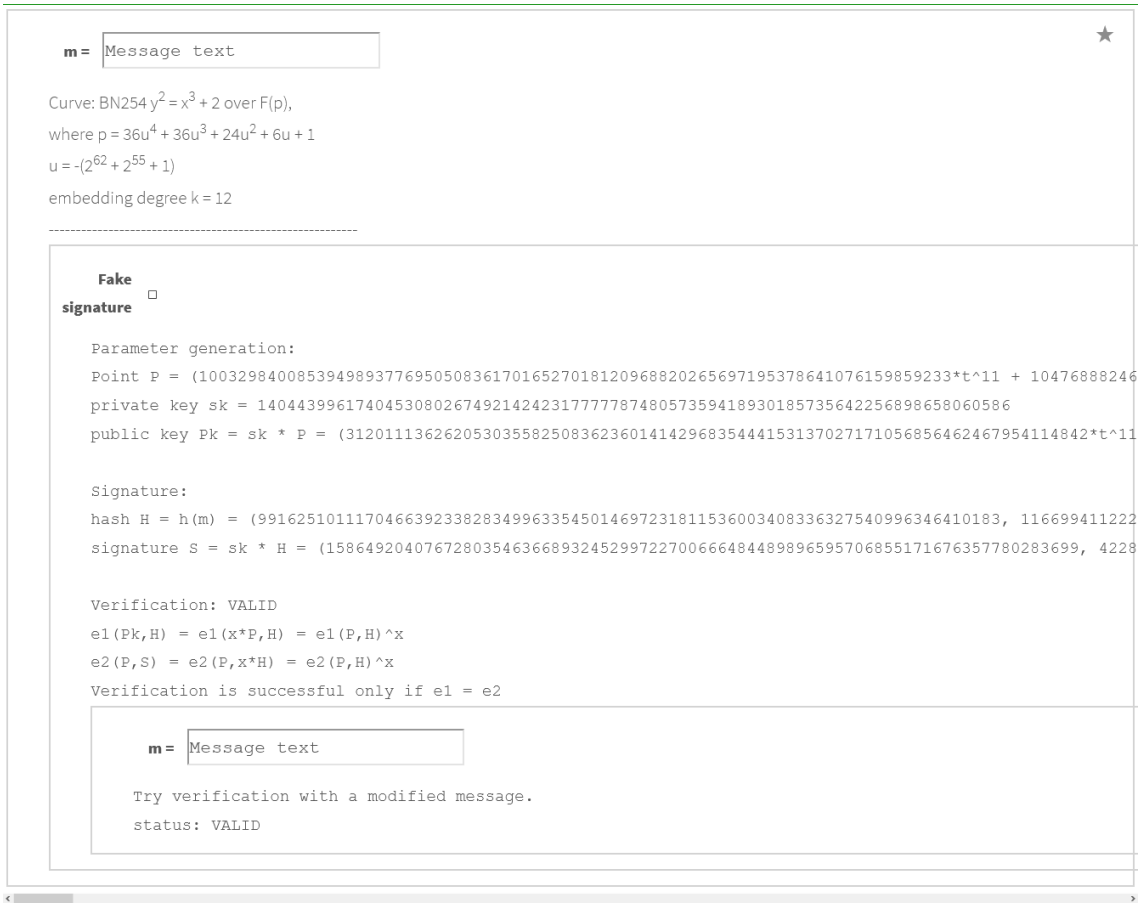

Fig. 3.10: BLS scheme

Conclusion

The application created as part of this thesis was initially developed using a local installation of Sage on the same server, where the webpage is hosted. To minimize the space required, as the size of installed Sage application is in the magnitude of GBs, a remote service called SageMathCell was used. SageMathCell provides the functionality of standard Sage installation, but serves as an interface. The computations are done remotely. Website it self is based on a template, which is free to use under a Creative Commons license. It uses HTML files, CSS files and JQuery for design elements.

The application is split into multiple parts, which can be accessed either from the tiles (clickable buttons) on the `index.html` page or from a drop down menu. Each of these parts represents an example of an operation on elliptic curves. They include displaying points on EC over a finite field and their plot in $\mathbb{F}_p$, plot of the curve in $\mathbb{R}$, performing point addition of two points on a user–selected curve in the generalized Weierstrass form and scalar multiplication. Bilinear pairings are also represented, users can compute Weil and Tate pairings on preselected curves. Further use of bilinear pairings comes into play with 3-way Diffie–Hellman key exchange and subsequently, MOV attack.

Bibliography

[1] WASHINGTON, Lawrence C. *Elliptic curves: Number theory and Cryptography*. 2nd ed. Boca Raton, FL: CRC Press, 2008. Discrete mathematics and its applications. ISBN 978-1420071467.

[2] HAJNÝ, J.; DZURENDA, P.; RICCI, S.; MALINA, L.; VRBA, K. *Performance Analysis of Pairing-Based Elliptic Curve Cryptography on Constrained Devices*. 2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2018, pp. 1-5, https://doi.org/10.1109/ICUMT.2018.8631228.

[3] LAURIDSEN, Martin M. *A Short Note on Cryptography using Elliptic Curves, Bilinear Pairings and Lattices* [online]. Technical University of Denmark, s. 1-36 [cit. 2021-5-28]. Dostupné z: https://martinlauridsen.info/pub/intro_to_pairings_lattices.pdf

[4] LANDWEBER, Peter S. *Elliptic Curves and Modular Forms in Algebraic Topology*: Proceedings of a Conference held at the Institute for Advanced Study, Princeton, Sept. 15-17, 1986. Springer-Verlag Berlin Heidelberg, 1988. Discrete mathematics and its applications. ISBN 978-3-540-39300-9.

[5] MULLEN, Gary L. and PANARIO, Daniel. *Handbook of Finite Fields*. Boca Raton, FL: CRC Press, 2013, 1068 s. Discrete Mathematics and Its Applications. ISBN 978-1439873786.

[6] BERNSTEIN, Daniel J. *Introduction to post-quantum cryptography*. In: Bernstein D.J., Buchmann J., Dahmen E. (eds) Post-Quantum Cryptography, 2009. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-88702-7_1

[7] SILVERMAN, J.H. and SUZUKI, J. *Elliptic Curve Discrete Logarithms and the Index Calculus*. In: Ohta K., Pei D. (eds) Advances in Cryptology — ASIACRYPT'98. ASIACRYPT 1998. Lecture Notes in Computer Science, vol 1514. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-49649-1_10

[8] MENEZES, Alfred J., Paul C. van OORSCHOT and Scott A. VANSTONE. *Handbook of applied cryptography*. Boca Raton: CRC, 1997. CRC Press series on discrete mathematics and its applications. ISBN 08-493-8523-7.

[9] GALBRAITH, Steven D.; PATERSON, Kenneth G.; SMART, Nigel P. *Pairings for cryptographers*. In: Discrete Applied Mathematics, Volume 156, Issue

16, 2008, p. 3113-3121, ISSN 0166-218X, https://doi.org/10.1016/j.dam.2007.12.010.

[10] MENEZES, A. *An Introduction to Pairing-Based Cryptography.* , UIMP-RSME Santaló Summer School. *Recent Trends in Cryptography.* Santander: American Mathematical Soc., 2009, s. 47-65. ISBN 978-0821839843.

[11] MOODY, D. *The Diffie–Hellman problem and generalization of Verheul's theorem.* Designs, Codes and Cryptography volume. 2009, 52(2), 381–390. Dostupné z: doi:https://doi.org/10.1007/s10623-009-9287-x

[12] CHENG, Z., VASIU, L., COMLEY, R. *Pairing-Based One-Round Tripartite Key Agreement Protocols.* Cryptology ePrint Archive: Report 2004/079. http://www.eprint.iacr.org/2004/079 (2004)

[13] BONEH D., LYNN B., SHACHAM H. *Short Signatures from the Weil Pairing.* In: Boyd C. (eds) Advances in Cryptology — ASIACRYPT 2001. ASIACRYPT 2001. Lecture Notes in Computer Science, vol 2248. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45682-1_30

[14] FLORIAN, L.; MIRELES, David J.; SHPARLINSKI, Igor E. *MOV attack in various subgroups on elliptic curves.* Illinois J. Math. 48 (3) 1041 - 1052, Fall 2004. https://doi.org/10.1215/ijm/1258131069

[15] *Sage Tutorial.* In: SageMath Documentation [online]. 2020 [cit. 2020-12-9]. Dostupné z: https://doc.sagemath.org/pdf/en/tutorial/SageTutorial.pdf

[16] *Embedding Sage Cells.* GitHub [online]. [cit. 2020-12-9]. Dostupné z: https://github.com/sagemath/sagecell/blob/master/doc/embedding.rst

[17] *Elliptic Curves.* SageMath Documentation [online]. 2020 [cit. 2020-12-9]. Dostupné z: https://doc.sagemath.org/html/en/constructions/elliptic_curves.html

[18] JANOUT, Vladimír. *Application for Elliptic Curve Cryptography.* Brno, 2020. Bachelor's thesis. Brno University of Technology, Faculty of Electrical Engineering and Communication. Vedoucí práce M.Sc. Sara Ricci, Ph.D.

List of symbols, quantities and abbreviations

BLS Boneh–Lynn–Shacham signature scheme
DH Diffe–Hellman protocol
DLP Discrete Logarithm Problem
EC Elliptic Curve
ECC Elliptic Curve Cryptography
ECDH Elliptic Curve Diffie–Hellman
ECDLP Elliptic Curve Discrete Logarithm Problem
HTML HyperText Markup Language
RSA Rivest, Shamir, Adleman cryptosystem

List of appendices

A Website tabs

A.1 Points on EC

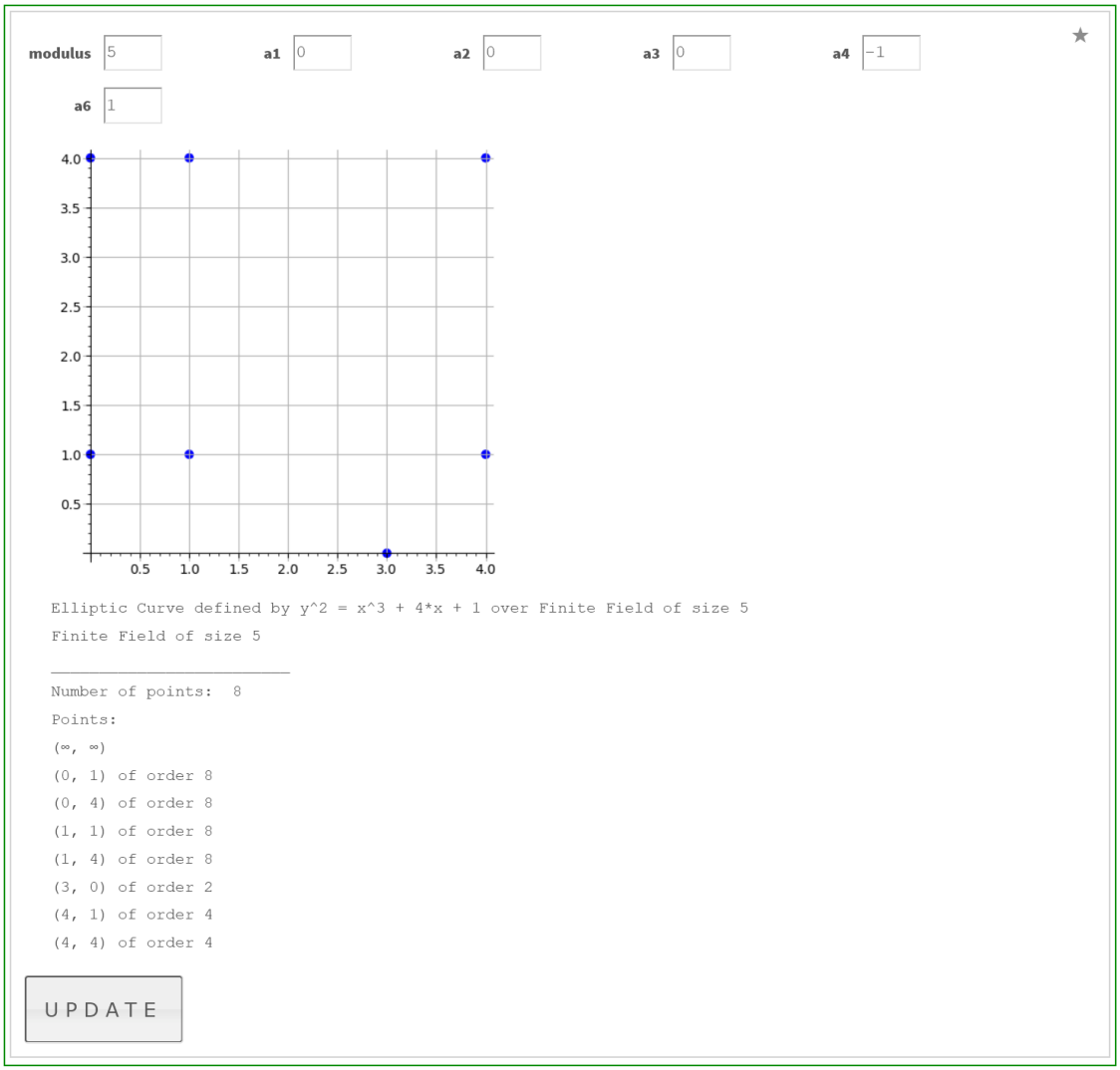

Fig. A.1: Points on EC

A.2 Point addition

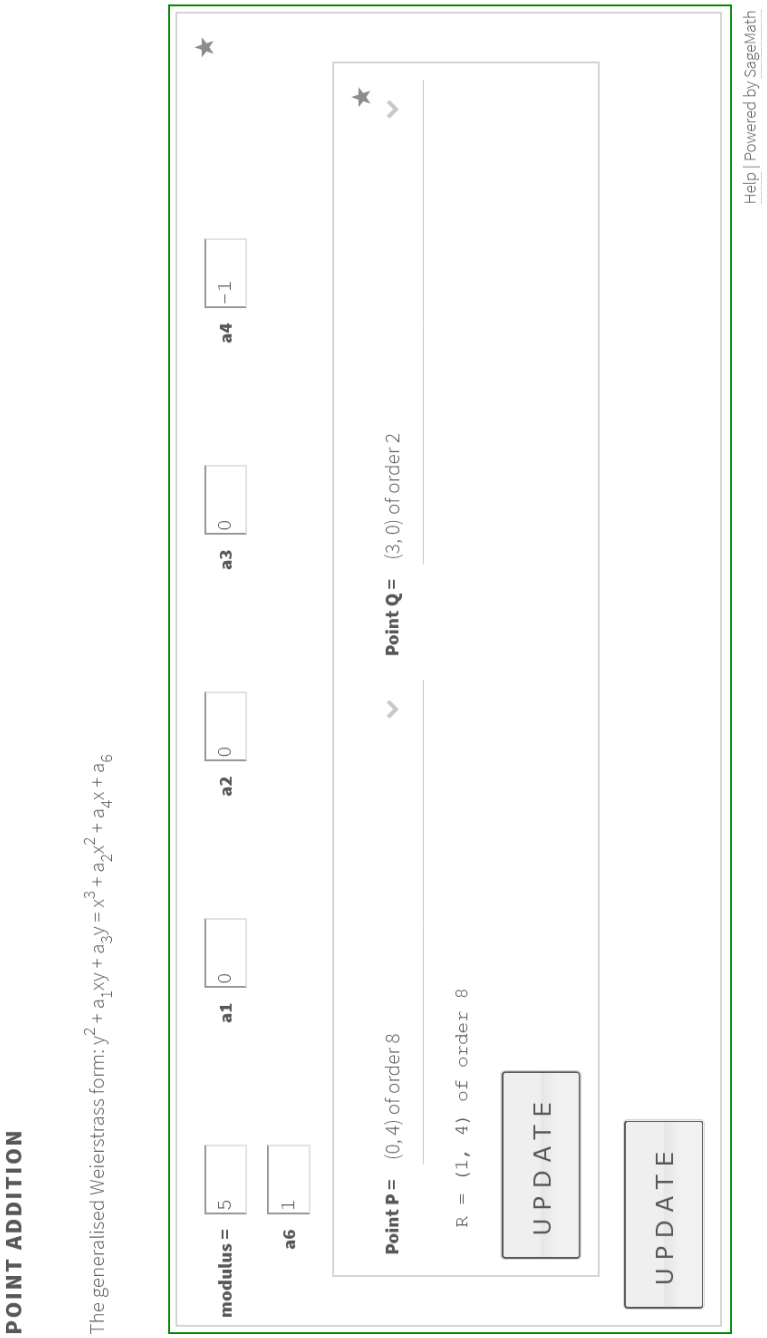

Fig. A.2: Point addition

A.3 Scalar multiplication

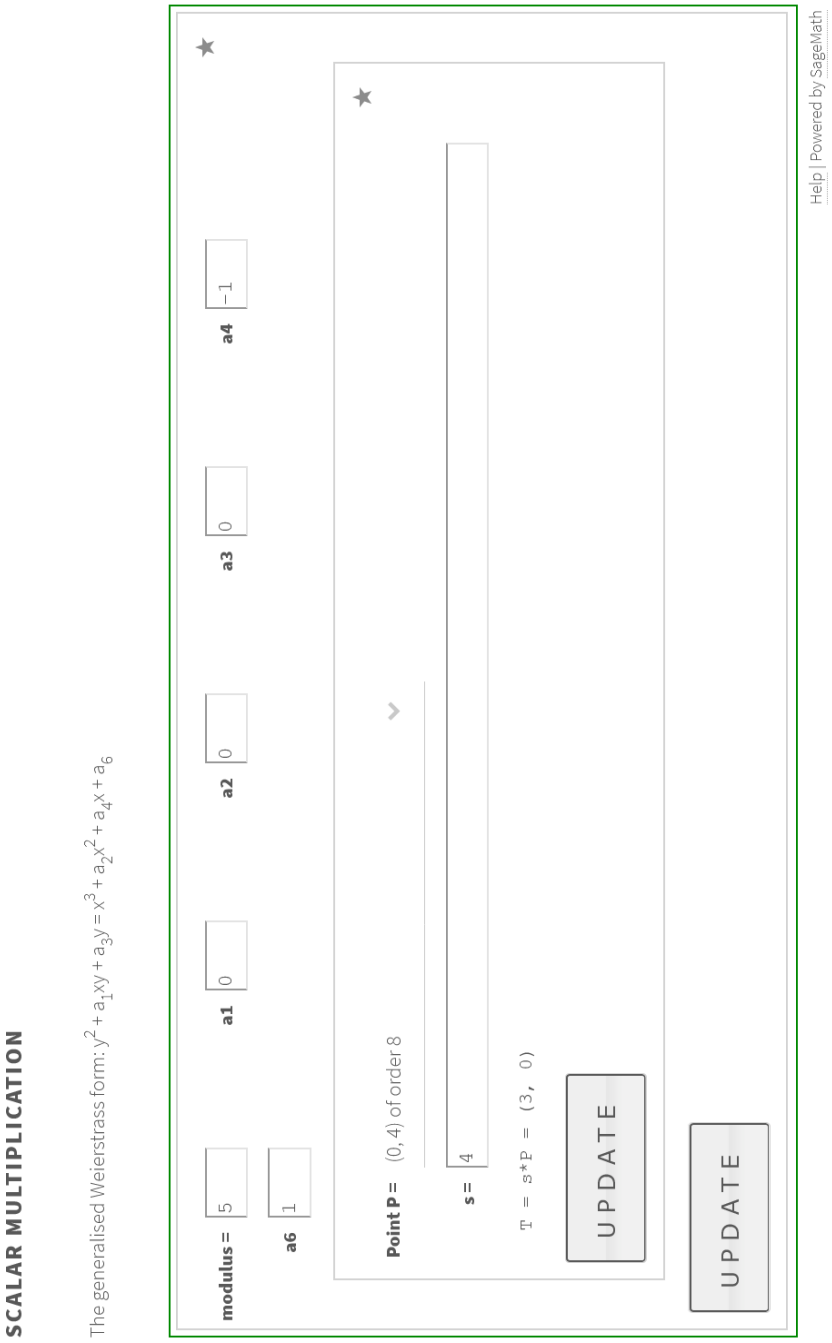

Fig. A.3: Scalar multiplication

A.4 Plot an EC

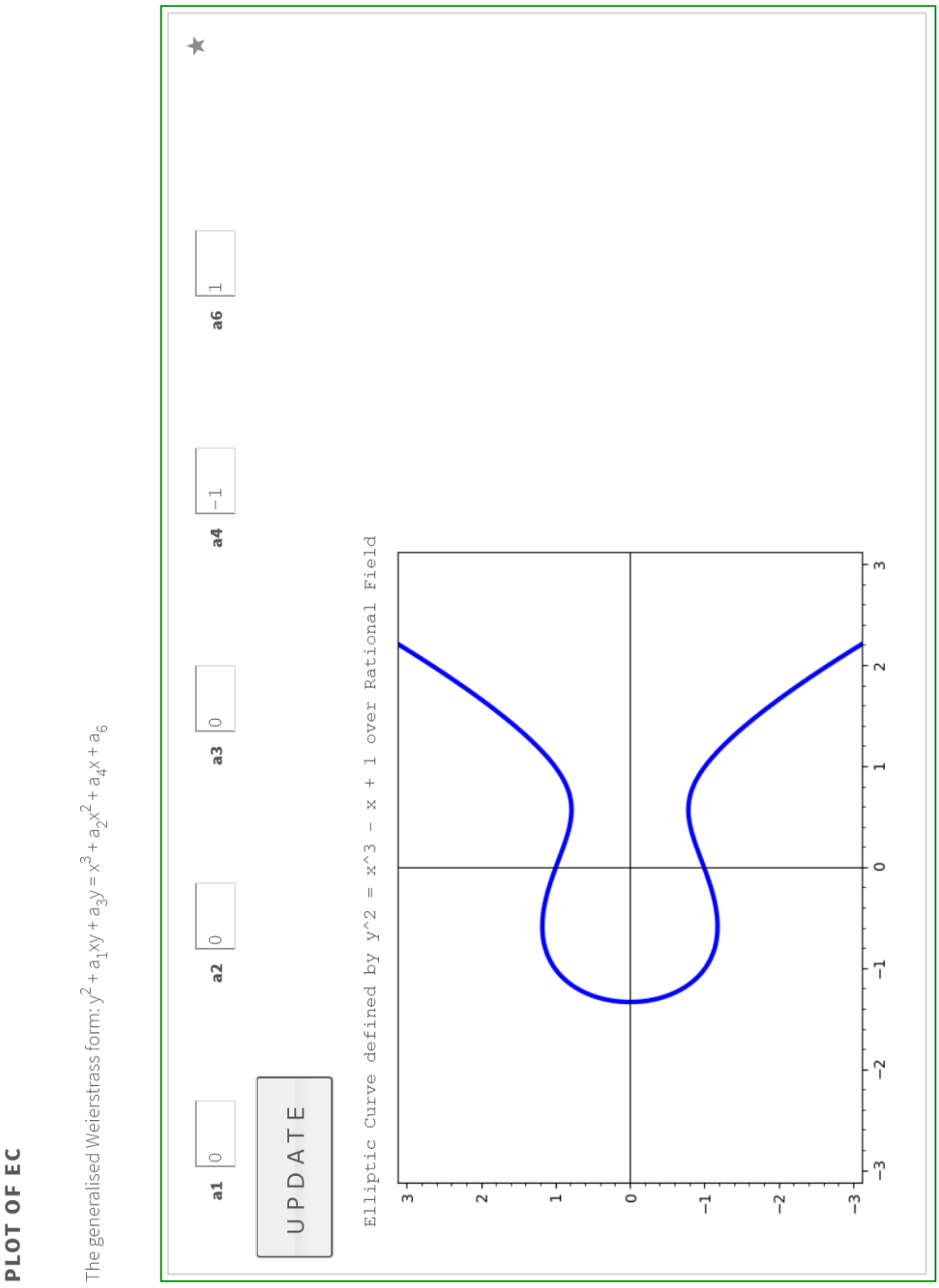

Fig. A.4: Plot an EC

A.5 Bilinear pairings

Fig. A.5: Weil pairing

TATE PAIRING

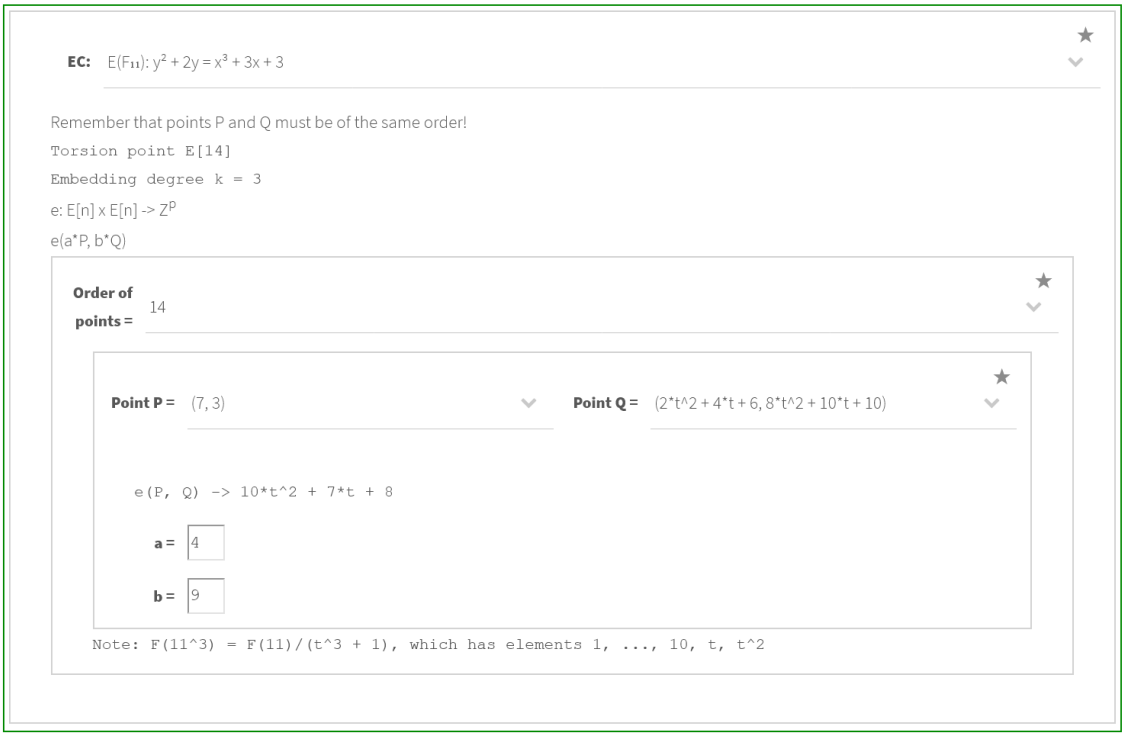

Fig. A.6: Tate pairing

A.6 3-way Diffie–Hellman Exchange

Fig. A.7: 3-way Diffie–Hellman Exchange

A.7 MOV attack

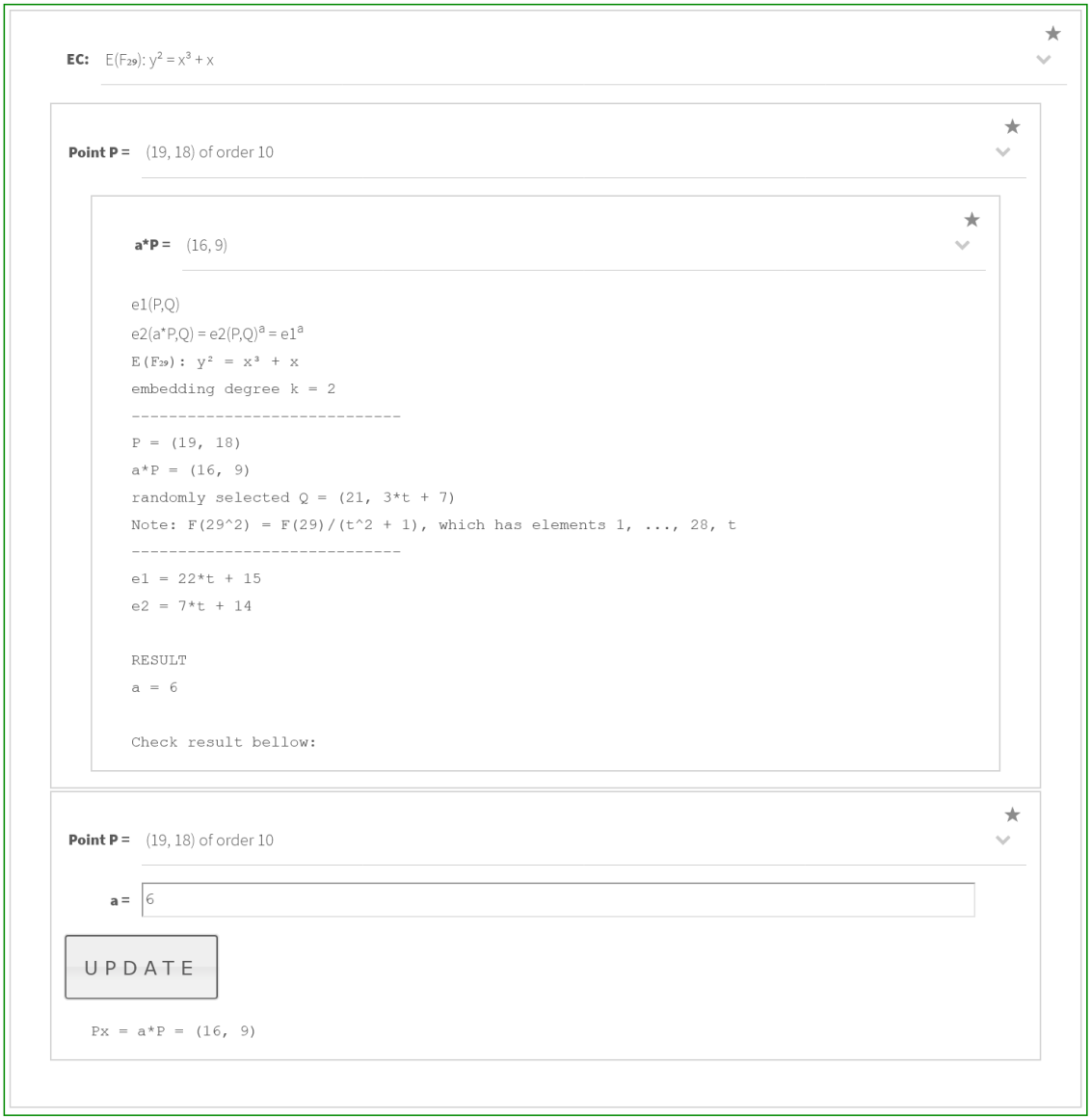

Fig. A.8: MOV attack

A.8 BLS signature scheme

Fig. A.9: BLS signature scheme