Discrete Logarithm Cryptography

by

Koray Karabina

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2010

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The security of many cryptographic schemes relies on the intractability of the discrete logarithm problem (DLP) in groups. The most commonly used groups to deploy such schemes are the multiplicative (sub)groups of finite fields and (hyper)elliptic curve groups over finite fields. The elements of these groups can be easily represented in a computer and the group arithmetic can be efficiently implemented.

In this thesis we first study certain subgroups of characteristic-two and characteristic-three finite field groups, with the goal of obtaining more efficient representation of elements and more efficient arithmetic in the corresponding groups. In particular, we propose new compression techniques and exponentiation algorithms, and discuss some potential benefits and applications.

Having mentioned that intractability of DLP is a basis for building cryptographic protocols, one should also take into consideration how a system is implemented. It has been shown that realistic (validation) attacks can be mounted against elliptic curve cryptosystems in the case that group membership testing is omitted. In the second part of the thesis, we extend the notion of validation attacks from elliptic curves to hyperelliptic curves, and show that singular curves can be used effectively in such attacks.

Finally, we tackle a specific location-privacy problem called the nearby friend problem. We formalize the security model and then propose a new protocol and its extensions that solve the problem in the proposed security model. An interesting feature of the protocol is that it does not depend on any cryptographic primitive and its security is primarily based on the intractability of the DLP. Our solution provides a new approach to solve the nearby friend problem and compares favorably with the earlier solutions to this problem.

Acknowledgments

I am grateful to Alfred Menezes for his supervision during my Ph.D. studies. His broad knowledge, encouraging support, generosity for making time for my questions on any matter, and his sense of humor made my life easier and my time enjoyable.

I thank my thesis defense committee members Ian Blake, Guang Gong, David Jao, Alfred Menezes and Doug Stinson for their valuable comments and questions.

I acknowledge David Wagner (University of Waterloo) for suggesting the discretization strategy used in the $\mathcal{NFP}$-II protocol.

I acknowledge the Department of Combinatorics and Optimization, NSERC and MITACS for the financial support they provided.

I thank Sanjit Chatterjee and Berkant Ustaoğlu with whom I enjoyed working on cryptography problems and having *non cryptical* conversations in our breaks.

To my parents İnci and Nihat, and my brothers İlkay and Olcay: thank you so much for your love and support during my time in Canada. Olcay, reading your e-mail first thing in the morning, receiving a call from you and planning things that we will do kept me super motivated.

My special thanks go to my darling, Burcu, who shared the excitement in my studies by all means and helped me overcome any obstacles I faced. I feel so lucky that you were always by me.

Sevgilim Burcu için...

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Discrete logarithm cryptography, in its broadest sense, is concerned with cryptographic schemes whose security relies on the intractability of the discrete logarithm problem (DLP), together with the underlying mathematical structures, implementation methods, performance/usability comparisons etc.

This thesis consists of three contributions in discrete logarithm cryptography. In the first part of the thesis, we study certain subgroups of characteristic-two and characteristic-three finite fields and propose new compression techniques and exponentiation algorithms that use compressed representation of elements in these groups. We discuss the advantages of using these techniques over the conventional techniques in discrete logarithm cryptography. In the second part, we re-emphasize the importance of validating public keys in discrete logarithm cryptosystems by extending the notion of validation attacks from elliptic curves to hyperelliptic curves, as well as by showing that singular curves can be used effectively in such attacks. In the last part, we propose a new protocol to solve the *nearby friend problem*; its efficiency is comparable to the earlier solutions and its security is based on the intractability of DLP. Our protocol does not depend on any cryptographic primitive as opposed to other solutions in the literature, which require for example the use of a homomomorphic encryption scheme.

In Sections 1.1, 1.2 and 1.3, we present a review of the literature, motivate the problems considered, and summarize our contributions on these three parts.

1.1 Compression in small characteristic finite fields

The Diffie-Hellman key agreement protocol [24] can be used by two parties A and B to establish a shared secret by communicating over an unsecured channel. Let $G = \langle g \rangle$ be a

prime-order subgroup of the multiplicative group $\mathbb{F}_q^*$ of a finite field $\mathbb{F}_q$. Party A selects a private key a and sends g^a to B. Similarly, B selects a private key b and sends g^b to A. Both parties can then compute the shared secret g^{ab}. Security of the protocol depends on the intractability of the problem of computing g^{ab} from g^a and g^b; this is called the Diffie-Hellman problem in G. The best method known for solving the Diffie-Hellman problem in G is to solve the discrete logarithm problem in G, that is, computing a from g^a. If q is prime (say $q = p$), then the fastest algorithms known for solving the discrete logarithm problem in G are Pollard's rho method [73] and the number field sieve [38]. To achieve a 128-bit security level against these attacks, one needs to select $\#G \approx 2^{256}$ and $p \approx 2^{3072}$ [31, Section 4.2]. Note that even though the order of G is approximately 2^{256}, the natural representation of elements of G, namely as integers modulo p, are approximately 3072 bits in length. If q is a power of 2 or 3, then one needs to select $\#G \approx 2^{256}$ and $q \approx 2^{4800}$ to achieve a 128-bit security level against Pollard's rho method and Coppersmith's index-calculus attack [20, 58]. This brings an overhead both to the efficiency of the protocol and to the number of bits that need to be stored or transmitted. In recent years, there have been several proposals for compressing the elements of certain subgroups of certain finite fields.

The first proposal was by Smith and Skinner in 1994 [83]; see also [10]. The main idea is that Lucas functions can be used modulo a prime to perform exponentiation in cryptographic applications. In fact, using this method the elements of the order-$(q+1)$ subgroup G of $\mathbb{F}_{q^2}^*$ can be identified by their traces over $\mathbb{F}_q$. More precisely, the elements of G can be uniquely identified up to conjugation over $\mathbb{F}_q$. This construction yields a compression factor of 2.

Gong and Harn [37] obtained a factor-3/2 compression and efficient exponentiation for the compressed form of the elements in the order-(p^2+p+1) subgroup G of $\mathbb{F}_{p^3}^*$. Elements of G are represented by a pair of elements from $\mathbb{F}_p$. Similarly, Giuliani and Gong [34] obtained a factor-5/2 compression and efficient exponentiation for the compressed form of the elements in the order-$(p^4-p^3+p^2-p+1)$ subgroup G of $\mathbb{F}_{p^{10}}^*$. Brouwer, Pellikaan and Verheul [13] obtained a factor-3 compression by representing elements of the order-(p^2-p+1) subgroup G of $\mathbb{F}_{p^6}^*$ by a pair of elements from $\mathbb{F}_p$. Even though they did not give an algorithm to exponentiate the elements in G in their compressed form, they noted that to exponentiate an element in G it suffices to know its compressed form and the exponent. In 2000, Lenstra and Verheul [59] showed that elements of the order-(p^2-p+1) subgroup G of $\mathbb{F}_{p^6}^*$ can be uniquely represented (up to conjugation over $\mathbb{F}_{p^2}$) by their traces over $\mathbb{F}_{p^2}$. Note that the compression factor is the same as in [13]. An important contribution of Lenstra and Verheul was a very efficient algorithm for exponentiation in G using the trace representation. More recently, Shirase et al. [78] observed that the elements of the order-$(q-\sqrt{3q}+1)$ subgroup G of $\mathbb{F}_{q^6}^*$, where $q = 3^m$ for some odd number m, can be uniquely represented (up to conjugation over $\mathbb{F}_q$) by their traces over $\mathbb{F}_q$, thereby achieving

a factor-6 compression. They also presented an algorithm for exponentiation in G. If $g \in G$ and c is its factor-6 compressed form in $\mathbb{F}_q$, they first lifted c to the trace of g in $\mathbb{F}_{q^2}$ and thereafter used an analogue of the Lenstra-Verheul algorithm to exponentiate g.

Rubin and Silverberg [74] introduced a compression/decompression method for finite field elements by using a rational parametrization of an algebraic torus. They observed that for a positive integer k and a prime power q, one can define an algebraic torus $\mathbb{T}_{q,k}$, a $\varphi(k)$-dimensional algebraic variety over $\mathbb{F}_q$, such that its group $\mathbb{T}_k(\mathbb{F}_q)$ of $\mathbb{F}_q$-rational points is isomorphic to the order-$\Phi_k(q)$ (cyclotomic) subgroup $G_{q,k}$ of $\mathbb{F}_{q^k}^*$. Here, $\Phi_k(q)$ is the kth-cyclotomic polynomial evaluated at q, and φ is Euler's totient function. For the cases $k = 2$ and $k = 6$, Rubin and Silverberg presented explicit compression/decompression algorithms for the elements of $\mathbb{T}_k(\mathbb{F}_q)$, and showed that the Smith-Skinner, Gong-Harn and Lenstra-Verheul representations are based on certain quotients of the algebraic tori, $\mathbb{T}_{q,2}, \mathbb{T}_{q,3}$ and $\mathbb{T}_{q,6}$, respectively, thus explaining the compression ratios of $2/\varphi(2) = 2$, $3/\varphi(3) = 3/2$ and $6/\varphi(6) = 3$. Later, van Dijk et al. [88], improving on an earlier work [89], constructed an efficient bijection between $\mathbb{T}_k(\mathbb{F}_q) \times \mathbb{F}_k^m$ and $\mathbb{F}_q^{\varphi(k)+m}$ and obtained, asymptotically, a compression factor of $k/\varphi(k)$. In particular, when representing i elements in $\mathbb{T}_k(\mathbb{F}_q)$ for $k = 30$ and $k = 210$, they obtained compression factors $30i/(8i + 2)$ with $m = 2$, and $210i/(48i + 24)$ with $m = 24$, respectively.

Having summarized the two different compression approaches in finite fields it is natural to ask the following two questions. First, what is the best possible compression ratio for the elements of a subgroup G of the multiplicative group $\mathbb{F}^*$ of a finite field $\mathbb{F}$, where $\mathbb{F}$ is the minimal field with $G \subset \mathbb{F}^*$? One should of course require the corresponding compression and decompression functions to be efficiently computable, and the decompression of an element to be unique or almost unique. Second, how does the torus-based compression method compare to the trace-based compression method? The first question was partially answered in [74] for cyclotomic subgroups of finite fields. As already mentioned above, Rubin and Silverberg noted that one would hope to use only $\varphi(k)$ elements in $\mathbb{F}_q$ in order to (uniquely) represent elements of $G_{q,k}$, which, in fact, seems to give the best possible compression factor as $|G_{q,k}| \approx q^{\varphi(k)}$. However, one might expect better compression factors when considering proper subgroups G of $G_{q,k}$. In general, it would be desirable to compress the elements of any order-ℓ subgroup $G_\ell \subsetneq G_{q,k} \subset \mathbb{F}_{q^k}^*$ by a factor $(k \log q)/\log \ell$ in any characteristic.

In the first part of the thesis, we achieve factor-4 and factor-6 compression of certain proper subgroups G of $G_{q,4}$ and $G_{q,6}$ in characteristic-two and characteristic-three finite fields, respectively. In particular, $|G| = q \pm \sqrt{2q} + 1$ in characteristic-two fields and $|G| = q \pm \sqrt{3q} + 1$ in characteristic-three fields. Note that these compression factors are twice the compression factors for groups $G_{q,4}$ and $G_{q,6}$, which seem to be the best possible compression factors since $|G| \approx q$.

More precisely, we propose two compression methods: trace-based compression and

torus-based compression. In Chapter 2, we discuss the trace-based compression and its applications, and present several exponentiation algorithms using trace-based compressed representation of elements. In Chapter 3, we show that double-exponentiation using trace-based compressed representation of elements can be performed efficiently. This speeds up the exponentiation algorithms given in Chapter 2, in addition to allowing us to deploy our compression technique in some other cryptographic applications. We present our torus-based compression technique in Chapter 4, together with several exponentiation algorithms using torus-based compressed representation of elements. Chapter 4 includes a comparison of the trace-based and torus-based compression techniques, and of the resulting exponentiation algorithms. We also argue in Chapter 4 that the torus-based compression techniques cannot, in general, be extended to achieve compression factor $(k \log q)/\log \ell$ for proper order-ℓ subgroups G_ℓ of $G_{q,k}$.

Sections 1.1.1, 1.1.2 and 1.1.3 provide more details on our contributions on compression.

1.1.1 Contributions of Chapter 2

Let $q = 2^m$, where m is odd, and note that

$$\begin{aligned} q^4 - 1 &= (q-1)(q+1)\Phi_4(q) \\ &= (q-1)(q+1)(q-\sqrt{2q}+1)(q+\sqrt{2q}+1). \end{aligned}$$

In Chapter 2 , we achieve a factor-4 compression for the subgroups G of $G_{q,4}$ of orders $q \pm \sqrt{2q} + 1$. We show that the elements of G can be uniquely represented (up to conjugation over $\mathbb{F}_q$) using their traces over $\mathbb{F}_q$, and that exponentiation in G can be efficiently performed using the compressed representations. Our method gives a better compression factor than the Smith-Skinner system that compresses elements in G by a factor of 2. We note that our factor-4 compression does not contradict the Rubin-Silverberg observation about the necessity of using $\varphi(k)$ $\mathbb{F}_q$-elements for representing elements of $G_{q,k}$ since our construction compresses elements of subgroups of relatively small order of $G_{q,k}$.

Let $q = 3^m$, where m is odd, and note that

$$\begin{aligned} q^6 - 1 &= (q^3-1)(q+1)\Phi_6(q) \\ &= (q^3-1)(q+1)(q-\sqrt{3q}+1)(q+\sqrt{3q}+1). \end{aligned}$$

As mentioned earlier, Shirase et al. showed that by using traces over $\mathbb{F}_q$ one can achieve a factor-6 compression (up to conjugation over $\mathbb{F}_q$) for the elements of the order-$(q-\sqrt{3q}+1)$ subgroup G of $G_{q,6}$. Moreover, exponentiation in G can be efficiently performed when elements are represented by their traces over $\mathbb{F}_{q^2}$. We observe that a similar compression technique and efficient exponentiation also applies to the order-$(q+\sqrt{3q}+1)$ subgroup G of

$G_{q,6}$. Suppose that $g \in G$ and $c \in \mathbb{F}_q$ is its factor-6 compressed representation. We present six exponentiation algorithms. The first works directly with the compressed element c. The second algorithm first lifts c to the trace of g over $\mathbb{F}_{q^3}$, and then employs an exponentiation algorithm of Scott and Barreto [77]. The third algorithm first lifts c to g, and then uses a conventional exponentiation method. In the fourth algorithm, we first determine $f_2(x)$, the minimal polynomial of g over $\mathbb{F}_{q^2}$, by partially decompressing c to an element in $\mathbb{F}_{q^2}$. Then we construct $\mathbb{F}_{q^6} = \mathbb{F}_{q^2}[x]/(f_2(x))$, and use a conventional exponentiation method. The idea of the fifth and sixth algorithms is similar to the fourth algorithm except that we use the minimal polynomials of g over $\mathbb{F}_{q^3}$ and $\mathbb{F}_q$, respectively. In the case where the base is fixed, the first algorithm is expected to be at least 54% faster than the XTR_3 algorithm presented in [78].

Besides reducing transmission costs in the Diffie-Hellman and related protocols, we observe that compression techniques have applications in pairing-based cryptography where bilinear pairings derived from supersingular elliptic curves of embedding degree 4 and 6 over finite fields $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$ are employed. The pairing values lie in prime-order subgroups of orders dividing $q \pm \sqrt{2q} + 1$ and $q \pm \sqrt{3q} + 1$ (where $q = 2^m$ or $q = 3^m$) of $\mathbb{F}^*_{2^{4m}}$ and $\mathbb{F}^*_{3^{6m}}$, respectively, and thus it can be beneficial to compress these pairing values.

1.1.2 Contributions of Chapter 3

Double-exponentiation in the context of compressed representations is the operation of computing the compressed representation of g^{ak+bl} given integers a, b, and the compressed representations of g^k, g^l. Having efficient double-exponentiation is crucial in cryptographic applications. For example, in ElGamal type signature schemes the verifier should perform a double-exponentiation to verify the signature on the received message. Moreover, double-exponentiation can be used to speed up single-exponentiation by representing the exponent $\tau = ak + b$ where k is some fixed integer and a, b are half the bitlength of τ. Then g^k can be precomputed and given τ, one can compute $g^\tau = (g^k)^a \cdot g^b$ using simultaneous exponentiation (*Straus-Shamir's trick*; see Algorithm 14.88 in [67]) much more efficiently than direct exponentiation by τ. As mentioned earlier, the trace function is not multiplicative and hence it is not clear if one can favorably exploit this idea when the trace representation of elements is used.

Lenstra and Stam [84] showed that in the case of factor-2 and factor-3 compression in large-prime characteristic fields one can perform double-exponentiation very efficiently and they discuss some related applications such as speeding up the single-exponentiation algorithm for compressed elements.

In Chapter 3, we show that double-exponentiation that works directly with factor-4 trace-based compressed representation of elements can be performed very efficiently in the

case of characteristic-two fields, and describe two particular cryptographic applications. As a first application, we show how to use double-exponentiation to speed up single-exponentiation thereby obtaining an estimated 20% acceleration over the fastest single-exponentiation algorithm presented in Chapter 2 when the base is general. Speeding up the single-exponentiation is important as it speeds up some cryptographic protocols using the factor-4 compression technique. For example, as we observe in Chapter 2, the factor-4 compression technique can be applied to the image of the symmetric bilinear pairing derived from an embedding degree $k = 4$ supersingular elliptic curve defined over a characteristic-two field. If this pairing is used to implement the identity-based key aggrement protocol of Scott [76], then the messages exchanged can be compressed by a factor of 4; moreover, the single-exponentiation in the protocol can be performed using the compressed representation of elements. As a second application, we give details on deploying factor-4 compressed representation of elements in the Nyberg-Rueppel signature scheme [72]; our method also reduces the size of public keys.

1.1.3 Contributions of Chapter 4

In Chapter 4, we first argue that the torus-based compression techniques cannot, in general, be extended to achieve compression factor $(k \log q)/\log \ell$ for proper order-ℓ subgroups G_ℓ of $G_{q,k}$. At first glance our arguments might appear to contradict the aforementioned compression factors 4 and 6 achieved in the case of $G_{q\pm\sqrt{2q}+1} \subset G_{q,4}$ where q is a power of 2, and $G_{q\pm\sqrt{3q}+1} \subset G_{q,6}$ where q is a power of 3. However, we explain why this discrepancy occurs, and how it helps to work in characteristic-two and three fields to compress the elements of certain subgroups G_ℓ by a factor $k \approx (k \log q)/\log \ell$, when $\ell \approx q$. In particular, we present torus-based compression methods in characteristic-two and three fields that achieve factor-4 and 6 compression, respectively. Our approach has the advantage that computing the decompression functions is essentially free. This yields more efficient exponentiation algorithms compared to the trace-based exponentiation algorithms where decompression is quite costly.

1.2 Validation in discrete logarithm cryptosystems

The purpose of public-key validation is to verify that a public key possesses certain arithmetic properties. Public-key validation is especially important in discrete logarithm protocols where a party $\hat{B}$ combines his private key with a public key received from a second party $\hat{A}$ to form a group element σ. A dishonest party $\hat{A}$ might select an invalid public key in such a way that the subsequent use of σ in the protocol leaks information about $\hat{B}$'s private key. Lim and Lee [62] demonstrated the importance of public-key validation

by presenting *small-subgroup attacks* on some discrete logarithm key agreement protocols that are effective if the receiver of a group element does not verify that the element belongs to the desired group of high order (e.g., a prime-order DSA-type subgroup of $\mathbb{F}_p^*$). In [9, 4], *invalid-curve attacks* were designed that are effective on elliptic curve protocols if the receiver of a point does not verify that the point indeed lies on the chosen elliptic curve; see also [64, 66, 65]. Chen, Cheng and Smart [19] illustrated the importance of public-key validation in identity-based key agreement protocols that use bilinear pairings.

1.2.1 Contributions of Chapter 5

The performance of low-genus hyperelliptic curves has been shown to be competitive with that of elliptic curves[1]; see [5] for a summary of recent work. We demonstrate that invalid-curve attacks can be successfully mounted on protocols based on genus-2 hyperelliptic curves if the appropriate public-key validation is not performed. We also show that *singular* curves can be used in mounting invalid-curve attacks against (hyper)elliptic curve protocols. We illustrate our attacks on two recently-proposed discrete logarithm protocols — the Twin Diffie-Hellman key agreement scheme [16] and the XCR signature scheme [55].

In order to analyze invalid-curve attacks on hyperelliptic curve cryptosystems, it is useful to know the number $N_g(q)$ of isomorphism classes of genus-g hyperelliptic curves over a finite field $\mathbb{F}_q$. The isomorphism classes of genus-g hyperelliptic curves over an algebraically closed field K are in 1-1 correspondence with the elements of a $(2g-1)$-dimensional irreducible subvariety H_g of the moduli space M_g over K (see [41, p.347]), suggesting that number is of the order of q^{2g-1}. This was confirmed by Nart [71], who gave a closed formula for $N_g(q)$. We give an elementary counting argument that $N_g(q) = 2q^{2g-1} + \mathcal{O}(gq^{2g-2})$.

1.3 Location privacy

The rapid advent of mobile computing and consequent introduction of new technologies like smart phones and GPS together with the ever increasing mobility of the human populace have opened up the possibility of a plethora of services that could not even be conceived of a decade earlier. Different kinds of location-based services are increasingly becoming popular among the users of mobile devices [90]. However, this also calls forth a concern about the privacy of the location information of the users of such devices. So one is confronted with two contradictory goals – users would like to take advantage of different services that cater to their needs based on a particular location while at the same time protecting the very

[1]Elliptic curves are the genus-1 hyperelliptic curves.

privacy of that location information. How to meet these apparently contradictory goals, or in other words to get the best of both worlds, is a major technological challenge.

In the last part of the thesis, we focus our attention on the following scenario. Suppose a user Alice, in a distributed mobile computing environment, wants to determine whether one of her friends, Bob, is in a nearby location or not. This is the so-called *nearby friend problem* and is relevant in the context of social networking and buddy tracking applications [3]. The trivial solution for Alice is to contact Bob and either reveal her own location to him or ask Bob to do so. But this comes at the cost of compromising one of the friend's location privacy altogether even when they are not actually nearby. Instead the users would like to have some control over their private location information so that Alice can learn Bob's location only if they are actually nearby and Bob is willing to reveal his location to Alice. The problem might be solved in a privacy-preserving way if we assume the existence of an *ideal* trusted third party with whom each party maintains a secret communication channel. Parties send their private location information to that trusted party who then decides whether they are nearby or not and informs them accordingly.

Based on the above discussion one may notice that the nearby friend problem is somewhat akin to the socialist millionaires' problem [45] in cryptology where the two participants learn whether they both have the same wealth or not without first disclosing their wealth. This can also be viewed as a concrete instantiation of the privacy-preserving set operations problem [52] or the two-party private matching problem [32]. However, the challenge here is to solve it in the resource-constrained environment of mobile devices.

We would like to solve the problem in a distributed setting without taking recourse to any trusted party and moreover solve it in such a way that the computational requirement, communication bandwidth, and number of communication rounds can be kept to a minimum. This, of course, has to be achieved in a secure way in an adversarial environment, i.e., the participants need the assurance that neither its communicating partner receives any information other than what (s)he is entitled to nor should it be possible for an eavesdropper to infringe upon their privacy.

Some initial attempts to solve the above problem can be found in the works of Atallah and Du [25] and Køien and Oleshchuk [54]. However, the solutions are less than satisfactory as the former requires a semi-trusted third party and several rounds of communications while the latter has some security vulnerabilities; see [94]. In a pioneering paper, Zhong, Goldberg and Hengartner [94] proposed three protocols to solve the above problem based on a cryptographic primitive called homomorphic encryption [36]. They assume that Alice and Bob establish a secure channel that provides both confidentiality and authentication prior to running the protocols. Such a communication channel can be established, for example, through a TLS connection [23]. Their first protocol, called Louis, requires the service of an online semi-trusted third party. The second protocol, called Lester, does not require any third party but has the disadvantage that, with some additional work, Alice might be

able to determine Bob's location even if they are not considered to be in nearby locations. The third protocol, called Pierre, does not suffer from any of the above deficiencies. It is quite efficient, in particular when one uses the homomorphic encryption scheme due to Cramer, Genarro and Schoenmakers [21] in the elliptic curve setting. Zhong [93] later proposed another protocol, called Wilfrid, based on the idea of private set intersection of [32]. This too makes use of homomorphic encryption and a communication channel that is both confidential and authenticated.

1.3.1 Contributions of Chapter 6

We take the work of Zhong et al. [94] as our point of departure by formalizing the statement of the nearby friend problem as well as its security model. We then proceed to propose a new protocol, called $\mathcal{NFP}$-I, in the line of Diffie-Hellman type of simple password exponential key exchange (SPEKE) [44]. An advantage of our protocol is that it only assumes the existence of an authenticated channel between the two parties, not a confidential one. Also note that we do not require any other cryptographic primitive such as homomorphic encryption that was used in [94] and [93]. The protocol is quite simple and we propose it as a new primitive that might serve as a building block for more complex privacy-preserving applications such as private matching [32]. We provide a security analysis to show that the protocol meets its desired objectives in the security model under the decision Diffie-Hellman assumption and a variant of it. Our protocol is quite efficient and compares favorably with the currently best protocol known, Wilfrid, that achieves the same functionality (see Table 1.1).

	Alice → Bob		Bob → Alice		decide whether nearby:
	# exps	message size	# exps	message size	# exps by Alice
Wilfrid	4	4	2.31	2	1
$\mathcal{NFP}$-I	1	1	2	2	1

Table 1.1: Performance comparison between Wilfrid and $\mathcal{NFP}$-I when implemented in a group G where $|G| \approx 2^{256}$ and the discrete logarithm problem is assumed to be hard. Both protocols require two communication steps. The message size is the number of elements of G.

We also discuss several extensions of this basic protocol $\mathcal{NFP}$-I. As a byproduct, we improve the functionality and the performance of the Pierre protocol by applying one of our extension techniques.

Chapter 2

Trace-Based Compression by a Factor-4 and 6

In this chapter we achieve trace-based factor-4 compression for the subgroups $G \subset \mathbb{F}_{q^4}^*$ of orders $q \pm \sqrt{2q} + 1$, where $q = 2^m$, m is odd; and factor-6 compression for the subgroups $G \subset \mathbb{F}_{q^6}^*$ of orders $q \pm \sqrt{3q} + 1$, where $q = 3^m$, m is odd. Moreover, we present and compare several algorithms for performing exponentiation in G using compressed representations. In particular, in the case where the base is fixed, we expect to gain at least a 54% speed up over the fastest previously known exponentiation algorithm that uses factor-6 compressed representations. The results of this chapter are to appear in [50].

The remainder of this chapter is organized as follows. We begin in Section 2.1 by describing some cryptographic applications of our compression methods. Section 2.2 introduces some terminology and sets the notation that we will use throughout Chapters 2 and 3. In Section 2.3, we provide some details on previous work. Sections 2.4 and 2.5 describe our compression and exponentiation techniques for characteristic-two and three fields. The exponentiation methods are compared in Section 2.6. We make some concluding remarks in Section 2.8.

2.1 Cryptographic applications

In this section we give some examples of cryptographic protocols where our compression techniques can be beneficial.

As described in Section 1.1, compression is useful in Diffie-Hellman and related key agreement protocols where the underlying group is a prime-order subgroup of the multiplicative group of a finite field. Indeed, Koblitz [53] studied the efficiency of discrete

logarithm protocols when the underlying group G is a subgroup of $\mathbb{F}_{q^4}^*$ for $q = 3^m$, and where $\#G$ divides $q \pm \sqrt{3q} + 1$.

Beginning with the seminal work of Joux [47], Sakai-Ohgishi-Kasahara [75] and Boneh-Franklin [11], bilinear pairings have been widely used to design protocols for various cryptographic tasks. These protocols can be described using symmetric bilinear pairings $e : G_1 \times G_1 \to G$ where G_1 and G are groups of prime order n. A necessary condition for the security of these protocols is that the discrete logarithm problems in G_1 and G should be intractable. Such pairings can be realized by selecting G_1 to be a group of points in $E_1(\mathbb{F}_q)$, where $q = 2^m$ and $E_1 : Y^2 + Y = X^3 + X + b$ where $b \in \{0, 1\}$ is a supersingular elliptic curve over $\mathbb{F}_q$ with $\#E_1(\mathbb{F}_q) = q \pm \sqrt{2q} + 1$. This elliptic curve has embedding degree 4, i.e., the smallest positive integer k for which $\#E_1(\mathbb{F}_q)$ divides $q^k - 1$ is $k = 4$. Then G is the order-n subgroup of $\mathbb{F}_{q^4}^*$. For example, if a 128-bit security level is desired, then one could use $E_1/\mathbb{F}_q : Y^2 + Y = X^3 + X$ with $q = 2^{1223}$ [2]. This curve has the property that $\#E_1(\mathbb{F}_q) = 5n$ where n is a 1221-bit prime, and so Pollard's rho method for solving the discrete logarithm problem in G_1 or G has running time approximately 2^{611}. Moreover, Coppersmith's algorithm [20] for solving the discrete logarithm problem in G has running time very roughly 2^{128} (see Table 6 of [58]).

Symmetric bilinear pairings can also be realized by selecting G_1 to be a group of points in $E_2(\mathbb{F}_q)$, where $q = 3^m$ and $E_2 : Y^2 = X^3 - X \pm 1$ is a supersingular elliptic curve over $\mathbb{F}_q$ with $\#E_2(\mathbb{F}_q) = q \pm \sqrt{3q} + 1$. This elliptic curve has embedding degree 6, and G is the order-n subgroup of $\mathbb{F}_{q^6}^*$. For example, if a 128-bit security level is desired, then one could use $E_2/\mathbb{F}_q : Y^2 = X^3 - X + 1$ with $q = 3^{509}$ [2]. This curve has the property that $\#E_2(\mathbb{F}_q) = 7n$ where n is a 804-bit prime, and so Pollard's rho method for solving the discrete logarithm problem in G_1 or G has running time approximately 2^{402}. Moreover, Coppersmith's algorithm for solving the discrete logarithm problem in G has running time very roughly 2^{128}.

In the Waters signature scheme [91], party A has a private key $Z = zP$ and a public key $\zeta = e(P, P)^z$. In order to sign a message M, A first computes $H = \text{Hash}(M) \in G_1$, where Hash is a cryptographic hash function that hashes its input elements into elliptic curve group elements, and chooses a random integer $r \in [1, n - 1]$. Then A computes $\alpha = Z + rH$ and $\beta = rP$ and sends (α, β) as her signature on M. A party B accepts A's signature on M if and only if $e(\alpha, P) = \zeta \cdot e(\beta, H)$. Our compression technique reduces the size of A's public key ζ by a factor of 4 or 6. More precisely, at the 128-bit security level, the size of the public key is reduced from 4892 bits to 1223 bits if E_1 is used, or from 4841 bits to 807 bits if E_2 is used. In order to verify A's signature, B can check if the compressed value of $e(\alpha, P)e(\beta, -H)$ is equal to the compressed value of ζ.

Another pairing-based application is the identity-based key agreement protocol of Scott [76]. In Scott's protocol, party A first computes a particular pairing value $g \in G$, and sends

$P_A = g^a$ to party B. Similarly, B computes the pairing value $g \in G$ and sends $P_B = g^b$ to A. Finally, both A and B compute the shared secret $P_B^a = P_A^b$. (For details of the computations, please refer to [76].) If the symmetric bilinear pairings described above are employed, then messages exchanged can be compressed by factors of 4 or 6, and moreover the computations can take place over smaller fields rather than $\mathbb{F}_{q^4}$ or $\mathbb{F}_{q^6}$.

2.2 Preliminaries and notation

Let q be a prime power, and let $\mathbb{F}_q$ denote a finite field with q elements. Let n be a prime such that $\gcd(n, q) = 1$, and let k be the smallest positive integer such that $q^k \equiv 1 \pmod{n}$. Then $\mathbb{F}_{q^k}$ has a multiplicative subgroup of order n which cannot be embedded in the multiplicative group of any extension field $\mathbb{F}_{q^i}$ for $1 \leq i < k$. For such a triple (q, k, n) we denote the multiplicative group of order n by μ_n and call k the *embedding degree* of μ_n over $\mathbb{F}_q$.

Let $g \in \mathbb{F}_{q^k}$ and let s be a positive divisor of k. We assume that g is not contained in any proper subfield of $\mathbb{F}_{q^k}$. The *conjugates* of g over $\mathbb{F}_{q^s}$ are $g_i = g^{q^{is}}$ for $0 \leq i < k/s$. The *trace* of g over $\mathbb{F}_{q^s}$ is the sum of the conjugates of g over $\mathbb{F}_{q^s}$, i.e.,

$$\mathrm{Tr}_s(g) = \sum_{i=0}^{\frac{k}{s}-1} g_i \in \mathbb{F}_{q^s}. \tag{2.2.1}$$

The *minimal polynomial* of g over $\mathbb{F}_{q^s}$ is the monic polynomial

$$f_{g,s}(x) = \prod_{i=0}^{\frac{k}{s}-1} (x - g_i). \tag{2.2.2}$$

Note that $f_{g,s}(x) \in \mathbb{F}_{q^s}[x]$. When $s = 1$ we simply use $\mathrm{Tr}(g)$ and $f_g(x)$ by abuse of notation. Also, we will assume that the conjugates of g over $\mathbb{F}_{q^s}$ are well defined for any integer i by setting $g_i = g_{i \bmod k/s}$.

We fix some notation for finite field operations that will be used in the remainder of the chapter. We will denote by $A_i, a_i, C_i, F_i, I_i, S_i, M_i$ and m_i the cost of operations of addition, addition by 1 or 2, cubing, exponentiation by a power of the characteristic of the field, inversion, squaring, multiplication, and multiplication by 2 in $\mathbb{F}_{q^i}$ for $i = 1, 2, 3$. $SR_{i,j}$ will denote the cost of finding a root of a degree i irreducible polynomial over $\mathbb{F}_{q^j}$. We use *soft-O* notation $\tilde{O}(\cdot)$ as follows: $a = \tilde{O}(b)$ if and only if for some constant c, $a = O(b(\log_2 b)^c)$.

2.3 Previous work on trace-based compression factors 2, 3 and 6

2.3.1 Compression factor 2 (LUC cryptosystem)

A Lucas sequence $V_i(P,Q)$ of the second kind is recursively defined by

$$V_0(P,Q) = 0,\ V_1(P,Q) = P,\ V_{i+2} = PV_{i+1}(P,Q) - QV_i(P,Q),$$

where P and Q are elements of a commutative ring with identity. In 1994, Smith and Skinner [83] proposed cryptosystems, which are analogous to Diffie-Hellman and ElGamal type cryptosystems, that use Lucas sequences $V_i(P,1)$ modulo a prime p (also see [70, 82] for earlier proposals that use Lucas sequences or Dickson polynomials modulo a composite number). In [83], it was argued that subexponential algorithms to compute discrete logarithms in finite fields cannot be applied to LUC cryptosystems as the Lucas sequences are not closed under multiplication. However, it was later shown that LUC cryptosytems are in fact reformulation of cryptosystems that work with the multiplicative subgroup of $\mathbb{F}_{p^2}^*$ of order $n = p+1$; see [10].

The key idea behind LUC cryptosystems is that Lucas functions can be used to perform *LUC-exponentiation* in a finite field. Noting that if $P \in \mathbb{F}_q$ and $x^2 - Px + 1 = (x-\alpha)(x-\beta)$ then $V_i(P,1) = \alpha^i + \beta^i$, the main idea behind LUC-exponentiation can be summarized as follows. Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}_{q^2}^*$ of order $n = q+1$. Let $c_u = \mathrm{Tr}(g^u)$ be the trace of g^u over $\mathbb{F}_q$, and $f_{g^u}(x)$ the minimal polynomial of g^u over $\mathbb{F}_q$. Then $f_{g^u}(x) = x^2 - c_u x + 1$ and the following recursive relation holds for all integers u and v:

$$c_{u+v} = c_u c_v - c_{u-v}.$$

In particular, $c_{2u} = c_u^2 - c_0 = c_u^2 - 2$ and $c_{2u+1} = c_{u+1}c_u - c_1$, whence given a state $s_u = [c_u, c_{u+1}]$ one can compute s_{2u} and s_{2u+1}. This observation leads to an efficient double-and-add algorithm for computing c_{ab} given c_a and b, and yields a compression factor of 2.

2.3.2 Compression factor 3 (LUCKY cryptosystem)

Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}_{p^6}^*$ of order $n = p^2 - p + 1$. Let $c_u = \mathrm{Tr}(g^u)$ be the trace of g^u over $\mathbb{F}_p$, and $f_{g^u}(x)$ the minimal polynomial of g^u over $\mathbb{F}_p$. Brouwer, Pellikaan and Verheul [13] obtained a compression factor 3 by identifying elements of μ_n with the coefficients of their minimal polynomials over $\mathbb{F}_p$. In particular, they showed that

$$f_{g^u}(x) = x^6 - c_u x^5 + d_u x^4 - e_u x^3 + d_u x^2 - c_u x + 1,$$

where

$$e_u = 2 + 2c_u + c_u^2 - 2d_u.$$

Hence, each element g^u can be uniquely identified (up to conjugation over $\mathbb{F}_p$) with a pair $(c_u, d_u) \in \mathbb{F}_p \times \mathbb{F}_p$. Brouwer, Pellikaan and Verheul also presented an algorithm (see Sections 3.3 and 5 in [13]) to compute (c_{ab}, d_{ab}) given (c_a, d_a) and an integer b. The algorithm is as follows. Given (c_a, d_a) we first construct the minimal polynomial of g^a over $\mathbb{F}_p$ and adjoin a root ρ_a of this polynomial to $\mathbb{F}_p$ thus obtaining a copy of $\mathbb{F}_{p^6}$. Next, we raise ρ_a to the power b and determine the minimal polynomial of ρ_a^b over $\mathbb{F}_p$. From the coefficients of this polynomial we can determine (c_{ab}, d_{ab}), as required. We note that the exponentiation algorithm is *commutative* and so it can be deployed in Diffie-Hellman-type protocols. That is, given (c_1, d_1) and integers a and b, if we compute (c_a, d_a) from (c_1, d_1) and a, and then compute (c_{ab}, d_{ab}) from (c_a, d_a) and b; and if we compute (c_b, d_b) from (c_1, d_1) and b, and then compute (c_{ba}, d_{ba}) from (c_b, d_b) and a, then we have $(c_{ab}, d_{ab}) = (c_{ba}, d_{ba})$. The proof follows from the following theorem.

Theorem 2.3.1. *Let $\mathbb{F}_1$ be a finite field of order p^6, and let $g \in \mathbb{F}_1$ be an element whose minimal polynomial $m_g(x)$ over $\mathbb{F}_p$ has degree 6. Let $\mathbb{F}_2 = \mathbb{F}[\alpha]/(m_g(\alpha))$, and let $\Psi : \mathbb{F}_1 \to \mathbb{F}_2$ be the isomorphism defined by $g \mapsto \alpha$. Finally, let $h \in \mathbb{F}_1$ and $\beta = \Psi(h)$. Then $m_h(x) = m_\beta(x)$.*

Proof. Since $m_h(h) = 0$, we have $\Psi(m_h(h)) = 0$. But $\Psi(m_h(h)) = m_h(\Psi(h)) = m_h(\beta)$, so $m_h(x)$ is the minimal polynomial of β over $\mathbb{F}_p$. □

2.3.3 Compression factor 3 (XTR cryptosystem)

Let $p \equiv 2 \pmod 3$ be a prime and $\mu_n = \langle g \rangle$ the multiplicative subgroup of $\mathbb{F}_{p^6}^*$ of order $n = p^2 - p + 1$. Let $c_u = \mathrm{Tr}(g^u)$ be the trace of g^u over $\mathbb{F}_{p^2}$, and $f_{g^u}(x)$ the minimal polynomial of g^u over $\mathbb{F}_{p^2}$. It was observed in [59] that $f_{g^u}(x) = x^3 - c_u x^2 + c_u^p x - 1$ and the following recursive relation holds for all integers u and v:

$$c_{u+v} = c_u c_v - c_v^p c_{u-v} + c_{u-2v}.$$

In particular, it was shown in [59] that given a state $s_u = [c_{u-1}, c_u, c_{u+1}]$ one can compute s_{2u} and s_{2u+1}. This observation leads to an efficient double-and-add algorithm for computing c_{ab} given c_a and b, and yields a compression factor of 3.

2.3.4 Compression factor 6 (XTR_3 cryptosystem)

Let $q = 3^{2r+1}$ and $\mu_n = \langle g \rangle$ the multiplicative subgroup of $\mathbb{F}_{q^6}^*$ of order $n = q - \sqrt{3q} + 1$. Let $c_u = \mathrm{Tr}(g^u)$ be the trace of g^u over $\mathbb{F}_q$. It was observed in [78] that given c_u one can

efficiently compute the trace $\tilde{c}_u$ of g^u over $\mathbb{F}_{q^2}$ up to conjugation over $\mathbb{F}_q$. Now, given c_a and b, one can first compute $\tilde{c}_a$, then compute $\tilde{c}_{ab}$ using an algorithm analogous to that of XTR, and finally obtain $c_{ab} = \tilde{c}_{ab} + \tilde{c}_{ab}^q$. This leads to an efficient double-and-add algorithm for computing c_{ab} given c_a and b, and yields a compression factor of 6.

2.4 Multiplicative groups with embedding degree 4

In this section we concentrate on multiplicative groups μ_n with embedding degree $k = 4$ over $\mathbb{F}_q$. In other words, we fix parameters (q, n) such that q is a prime power, n is a prime, $\gcd(q, n) = 1$, $q^4 \equiv 1 \pmod{n}$, and $q^i \not\equiv 1 \pmod{n}$ for $1 \leq i < 4$. Finally, we let h be a positive integer and define $t_h = q + 1 - h \cdot n$ to be the *trace* of μ_n over $\mathbb{F}_q$ with respect to the *cofactor* h. Throughout the rest of this section we will assume that the cofactor h is fixed, and we simply denote the trace of μ_n by t instead of t_h.

In the following lemma we show that $g \in \mu_n$ together with its conjugates can be uniquely represented by the pair $(\mathrm{Tr}(g), \mathrm{Tr}(g^t))$ of $\mathbb{F}_q$-elements. This already gives us compression by a factor 2. Furthermore, we will show in Corollary 2.4.5 that in characteristic-two finite fields it is possible to write all the coefficients of the minimal polynomial of g over $\mathbb{F}_q$ in terms of $\mathrm{Tr}(g)$ alone and hence achieve compression by a factor 4.

Lemma 2.4.1. *Let μ_n be the multiplicative subgroup of $\mathbb{F}_{q^4}^*$ of order n with embedding degree 4, trace t, and cofactor h. Let $g \in \mu_n$ and let g_i, for $i = 0, 1, 2, 3$, be the conjugates of g over $\mathbb{F}_q$. (Recall the convention that $g_i = g_{i \bmod 4}$.) Then*
(i) $g_i g_{i+1} = g_i^t$ for $i = 0, 1, 2, 3$.
(ii) $g_i g_{i+2} = 1$ for $i = 0, 1$.
(iii) $f_g(x) = x^4 - \mathrm{Tr}(g)x^3 + (\mathrm{Tr}(g^t) + 2)x^2 - \mathrm{Tr}(g)x + 1$.

Proof. (i) $g_i g_{i+1} = g_i^{q+1} = g_i^t$ since g_i is of order n and $q + 1 - t \equiv 0 \pmod{n}$.
(ii) $g_i g_{i+2} = g_i^{q^2+1} = 1$ since g_i is of order n and $q^2 + 1 \equiv 0 \pmod{n}$.
(iii) Using (i) and (ii) gives

$$\begin{aligned} f_g(x) &= \prod_{i=0}^{3}(x - g_i) \\ &= x^4 - \left(\sum_{i=0}^{3} g_i\right) x^3 + \left(\sum_{0 \leq i < j \leq 3} g_i g_j\right) x^2 - \left(\sum_{0 \leq i < j < k \leq 3} g_i g_j g_k\right) x + 1 \\ &= x^4 - \mathrm{Tr}(g)x^3 + (\mathrm{Tr}(g^t) + 2)x^2 - \mathrm{Tr}(g)x + 1. \end{aligned}$$

□

Suppose we fix a generator $g \in \mathbb{F}_{q^4}$ of μ_n, that is $\mu_n = \langle g \rangle$. In order to simplify the notation we define $c_u = \mathrm{Tr}(g^u)$ for any integer u. Note that $c_0 = 0$, $c_1 = \mathrm{Tr}(g)$, $c_u = c_{u \bmod n}$ and $c_{qu} = c_u^q = c_u$.

Lemma 2.4.2. *Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}_{q^4}^*$ of order n with embedding degree $k = 4$ and trace t. Then for all integers u and v we have*
(i) $c_u = c_{-u}$.
(ii) $c_u c_v = c_{u+v} + c_{u-v} + c_{u+v(t-1)} + c_{v+u(t-1)}$.

Proof. (i) This follows from Lemma 2.4.1(ii).
(ii) By Lemma 2.4.1(i) and (ii) it follows that

$$\begin{aligned}
c_u c_v &= (g_0^u + g_1^u + g_2^u + g_3^u)(g_0^v + g_1^v + g_2^v + g_3^v) \\
&= \sum_{i=0}^{3} g_i^{u+v} + \sum_{i=0}^{3} g_i^u g_{i+1}^v + \sum_{i=0}^{3} g_i^u g_{i+2}^v + \sum_{i=0}^{3} g_i^u g_{i+3}^v \\
&= c_{u+v} + \sum_{i=0}^{3} g_i^{u-v}(g_i^v g_{i+1}^v) + \sum_{i=0}^{3} g_i^{u-v}(g_i^v g_{i+2}^v) + \sum_{i=0}^{3} g_i^v g_{i+1}^u \\
&= c_{u+v} + \sum_{i=0}^{3} g_i^{u+v(t-1)} + \sum_{i=0}^{3} g_i^{u-v} + \sum_{i=0}^{3} g_i^{v+u(t-1)} \\
&= c_{u+v} + c_{u+v(t-1)} + c_{u-v} + c_{v+u(t-1)}.
\end{aligned}$$

□

Theorem 2.4.3. *Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}_{q^4}^*$ of order n with embedding degree 4 and trace t. Then for all integers u and v we have*

$$c_{u+v} = c_u c_v - c_{u-v}(c_{tv} + 2) + c_{u-2v} c_v - c_{u-3v}. \tag{2.4.1}$$

Proof. First start with

$$\begin{aligned}
c_u c_v &= (g_0^u + g_1^u + g_2^u + g_3^u)(g_0^v + g_1^v + g_2^v + g_3^v) \\
&= c_{u+v} + \sum_{i=0}^{3} \sum_{\substack{j=0 \\ j \neq i}}^{3} g_i^u g_j^v \\
&= c_{u+v} + \sum_{i=0}^{3} \sum_{\substack{j=0 \\ j \neq i}}^{3} g_i^{u-v}(g_i^v g_j^v) \\
&= c_{u+v} + \sum_{i=0}^{3} g_i^{u-v}(g_i^{tv} + g_{i+3}^{tv} + 1), \text{ by Lemma 2.4.1(i) and (ii)}
\end{aligned}$$

$$= \quad c_{u+v} + c_{u-v} + c_{u-v}c_{tv} - \sum_{i=0}^{3} g_i^{u-v}(g_{i+1}^{tv} + g_{i+2}^{tv}). \tag{2.4.2}$$

Then observe that

$$\begin{aligned}\sum_{i=0}^{3} g_i^{u-v} g_{i+1}^{tv} &= \sum_{i=0}^{3} g_i^{u-v} g_i^{qtv} \\ &= \sum_{i=0}^{3} g_i^{(u-2v)+qv} \text{ since } qt \equiv q^2 + q \equiv q - 1 \pmod{n}\end{aligned} \tag{2.4.3}$$

and

$$\begin{aligned}\sum_{i=0}^{3} g_i^{u-v} g_{i+2}^{tv} &= \sum_{i=0}^{3} g_i^{u-v} g_i^{-tv} \\ &= \sum_{i=0}^{3} g_i^{(u-2v)-qv}, \text{ since } t \equiv q + 1 \pmod{n}.\end{aligned} \tag{2.4.4}$$

Substituting (2.4.3) and (2.4.4) into (2.4.2) we obtain

$$c_u c_v = c_{u+v} + c_{u-v} + c_{u-v}c_{tv} - (c_{(u-2v)+qv} + c_{(u-2v)-qv}). \tag{2.4.5}$$

Now, let $u' = u - 2v, v' = qv$. Then

$$\begin{aligned} u' + v'(t-1) &\equiv (u-2v) + q^2 v \equiv u - 3v \pmod{n}, \\ v' + u'(t-1) &\equiv qv + (u-2v)q \equiv q(u-v) \pmod{n}, \end{aligned}$$

and using Lemma 2.4.2 (ii) with u', v' gives

$$\begin{aligned} c_{(u-2v)+qv} + c_{(u-2v)-qv} &= c_{u-2v}c_{qv} - (c_{u-3v} + c_{q(u-v)}) \\ &= c_{u-2v}c_v - (c_{u-3v} + c_{u-v}). \end{aligned} \tag{2.4.6}$$

Finally, (2.4.5) and (2.4.6) complete the proof. □

2.4.1 Characteristic-two finite fields

Let r be a positive integer, and let $q = 2^{2r+1}$, $t = \pm 2^{r+1}$, $T = |t|$. The values of r for which $q + 1 - t = hn$ and n is prime lead to a multiplicative subgroup μ_n of $\mathbb{F}_{q^4}^*$ of prime order n with embedding degree 4. Throughout this section we fix h, n, q, t, T and $\mu_n = \langle g \rangle$ in this way, and also write $c_u = \mathrm{Tr}(g^u)$.

The following recursive relations follow from Theorem 2.4.3 by noting that the characteristic of $\mathbb{F}_q$ is 2 and also that $c_{vt} = c_{vT} = c_v^T$ (see Lemma 2.4.2(i)).

Corollary 2.4.4. *Let μ_n be the multiplicative subgroup of $\mathbb{F}_{q^4}^*$ with embedding degree 4 and trace t. Then for all integers u and v we have*
(i) $c_{u+v} = c_u c_v + c_{u-v} c_v^T + c_{u-2v} c_v + c_{u-3v}$.
(ii) $c_{2u} = c_u^2$.

Corollary 2.4.5. *Let μ_n be the multiplicative subgroup of $\mathbb{F}_{q^4}^*$ with embedding degree 4 and trace t. Let $f_{g^u}(x)$ be the minimal polynomial of $g^u \in \mu_n$ over $\mathbb{F}_q$. Then*

$$f_{g^u}(x) = x^4 + c_u x^3 + c_u^T x^2 + c_u x + 1.$$

Proof. The proof follows from Lemma 2.4.1(iii) and Corollary 2.4.4(ii). □

Remark 2.4.6. Throughout the remainder of this section we will assume without loss of generality that the trace t is positive. If t is negative then one can replace the expressions of the form $c_u^{p(t)}$, where p is some polynomial, by $c_u^{p(T)}$ without changing the validity of the results in this section.

2.4.2 An exponentiation algorithm in μ_n

Corollary 2.4.5 shows that the element $g^u \in \mu_n$ can be represented uniquely (up to conjugation) by its trace c_u. Our next objective is to develop an efficient method for computing c_a given c_1 and a; this is the exponentiation operation in μ_n. We define $s_1 = [c_{-1}, c_0, c_1, c_2] = [c_1, 0, c_1, c_1^2]$ to be the initial state. For a given state $s_u = [c_{u-2}, c_{u-1}, c_u, c_{u+1}]$ with $u \geq 1$, if we can efficiently compute the states $s_{2u} = [c_{2u-2}, c_{2u-1}, c_{2u}, c_{2u+1}]$ and $s_{2u+1} = [c_{2u-1}, c_{2u}, c_{2u+1}, c_{2u+2}]$ then we immediately have an efficient double-and-add algorithm for computing c_a given c_1 and a.

Theorem 2.4.7. *Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}_{q^4}^*$ with embedding degree 4 and trace t. Let $c_u = \mathrm{Tr}(g^u)$,*

$$A = \begin{pmatrix} c_1 & c_1 & 0 & 0 \\ 0 & c_1 & c_1 & 0 \\ 0 & 1 & c_1^t & 1 \\ 1 & c_1^t & 1 & 0 \end{pmatrix}, \; X = \begin{pmatrix} c_{2u-3} \\ c_{2u-1} \\ c_{2u+1} \\ c_{2u+3} \end{pmatrix}, \; Y = \begin{pmatrix} (c_u + c_{u-2})^2 + c_{u-1}^2 c_1^t \\ (c_{u+1} + c_{u-1})^2 + c_u^2 c_1^t \\ (c_u + c_{u+1})^2 c_1 \\ (c_{u-1} + c_u)^2 c_1 \end{pmatrix}.$$

Then
(i) A is invertible and $AX = Y$.
(ii) If c_1 is given then A and A^{-1} can be efficiently computed.
(iii) $c_{2u-1} = \frac{1}{c_1^{t+1}} \left((c_{u+1} + c_u + c_{u-1} + c_{u-2})^2 + (c_u + c_{u-1})^2 (c_1^t + c_1^2) \right)$.
(iv) $c_{2u+1} = c_{2u-1} + \frac{1}{c_1} \left((c_{u+1} + c_{u-1})^2 + c_u^2 c_1^t \right)$.

Proof. (i) Noting that the characteristic of $\mathbb{F}_q$ is 2, we can show that the determinant of A is equal to c_1^{t+2}. Hence A is invertible if and only if $c_1 \neq 0$. In fact, c_1 is never zero as otherwise the minimal polynomial $f_g(x) = x^4 + 1 = (x+1)^4$ is not irreducible. This proves the first part. For the second part we combine the four equations obtained from Corollary 2.4.4 with the following (u, v) values: $(2u-3, -1), (2u-2, -1), (2u-1, -1), (2u, -1)$, and also note that $c_{2u} = c_u^2$.
(ii) Let $A^{-1}[i]$ be the ith row of A^{-1}. Then one can check that

$$\begin{aligned} A^{-1}[1] &= [(c_1^t+1)/c_1^{t+1}, 1/c_1^{t+1}, 0, 1/c_1^t], \\ A^{-1}[2] &= [1/c_1^{t+1}, 1/c_1^{t+1}, 0, 1/c_1^t], \\ A^{-1}[3] &= [1/c_1^{t+1}, (c_1^t+1)/c_1^{t+1}, 0, 1/c_1^t], \\ A^{-1}[4] &= [(c_1^t+1)/c_1^{t+1}, (c_1^{2t}+c_1^t+1)/c_1^{t+1}, 1, (c_1^{t+1}+1)/c_1^t] \end{aligned}$$

and the proof follows.
(iii) The inner product of $A^{-1}[2]$ and Y is equal to c_{2u-1}, and by part (ii) we can write

$$A^{-1}[2]Y = \frac{1}{c_1^{t+1}} \left((c_{u+1} + c_u + c_{u-1} + c_{u-2})^2 + (c_u + c_{u-1})^2 (c_1^t + c_1^2) \right).$$

(iv) The proof is similar to the proof of part (iii). □

The formulas for c_{2u-1} and c_{2u+1} in Theorem 2.4.7 yield Algorithm 2.1 for exponentiation in μ_n.

Remark 2.4.8. Algorithm 2.1 can be used to compute c_{ab} given c_a and b as follows. We set $c'_1 = c_a$ and the initial state becomes $s'_1 = [c'_{-1}, c'_0, c'_1, c'_2] = [c_a, 0, c_a, c_a^2]$. With input c'_1 and b, Algorithm 2.1 outputs $c'_b = c_{ab}$.

Since the cost of addition is negligible in finite fields of characteristic-two, we will ignore addition costs in the performance analysis of algorithms in this section. Moreover, we may ignore the cost $(1F_1 + 1S_1)$ in the precomputation steps of Algorithm 2.1 as it is negligible compared to $(1I_1 + 1M_1)$. Then the cost of Algorithm 2.1 can be approximated as:

Precomputation (steps 2 and 3): $1I_1 + 1M_1$.

Main loop (steps 4–15): $(4M_1 + 4S_1)(\ell - 1)$.

We note that Algorithm 2.1 has a limited degree of built-in resistance to side-channel analysis attacks because the same types of operations are executed whether the bit a_i of the exponent is 1 or 0.

Algorithm 2.1 A single-exponentiation algorithm
Input: c_1 and a
Output: c_a

1: Write $a = \sum_{i=0}^{\ell-1} a_i 2^i$ where $a_i \in \{0, 1\}$ and $a_{\ell-1} = 1$
2: $s_u = [c_{u-2}, c_{u-1}, c_u, c_{u+1}] \leftarrow [c_1, 0, c_1, c_1^2]$
3: $m_1 \leftarrow 1/c_1^{t+1}$ and $m_2 \leftarrow 1/c_1$
4: **for** i from $\ell - 2$ down to 0 **do**
5: $\quad c_{2u-1} \leftarrow m_1 \left((c_{u+1} + c_u + c_{u-1} + c_{u-2})^2 + (c_u + c_{u-1})^2 (c_1^t + c_1^2)\right)$
6: $\quad c_{2u} \leftarrow c_u^2$
7: $\quad c_{2u+1} \leftarrow c_{2u-1} + m_2 \left((c_{u+1} + c_{u-1})^2 + c_u^2 c_1^t\right)$
8: $\quad$ **if** $a_i = 1$ **then**
9: $\quad\quad c_{2u+2} \leftarrow c_{u+1}^2$
10: $\quad\quad s_u \leftarrow [c_{2u-1}, c_{2u}, c_{2u+1}, c_{2u+2}]$
11: $\quad$ **else**
12: $\quad\quad c_{2u-2} \leftarrow c_{u-1}^2$
13: $\quad\quad s_u \leftarrow [c_{2u-2}, c_{2u-1}, c_{2u}, c_{2u+1}]$
14: $\quad$ **end if**
15: **end for**
16: Return (c_u)

2.4.3 Other algorithms for exponentiation with compressed elements

Algorithm 2.1 works directly with the factor-4 compressed elements. In this section, we describe four algorithms for computing c_{ab} given c_a and b. The first algorithm partially decompresses c_a to an element $\tilde{c}_a \in \mathbb{F}_{q^2}$, and then uses the LUC method for exponentiating in this representation. The second algorithm decompresses c_a to an element in $\mathbb{F}_{q^4}$, and then employs a standard window-NAF exponentiation method. The third and fourth methods use the Brouwer-Pellikaan-Verheul idea (cf. Section 2.3.2) by using minimal polynomials over $\mathbb{F}_{q^2}$ and $\mathbb{F}_q$, respectively. The five exponentiation algorithms are compared in Section 2.6.

First, we prove the following.

Lemma 2.4.9. *Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}_{q^4}^*$ with embedding degree 4, trace t, and cofactor h. Let $c_u = \mathrm{Tr}(g^u)$ and $\tilde{c}_u = \mathrm{Tr}_2(g^u)$. Then $\{\tilde{c}_u, \tilde{c}_u^q\}$ is the set of roots of the polynomial $\tilde{f}_{g^u}(x) = x^2 + c_u x + c_u^t$.*

Proof. Since g has order n and $q + 1 \equiv t \pmod{n}$ and $q^2 \equiv -1 \pmod{n}$, we have $(x - \tilde{c}_u)(x - \tilde{c}_u^q) = (x - (g^u + g^{uq^2}))(x - (g^{uq} + g^{uq^3})) = x^2 + c_u x + c_u^t = \tilde{f}_{g^u}(x)$. □

An algorithm based on LUC-exponentiation

We first describe an algorithm to compute c_a given c_1 and a. The idea of the algorithm is as follows. Let $\tilde{d}_1 = \tilde{c}_1^q$. Suppose we know an element in the set $\{\tilde{c}_1, \tilde{d}_1\}$. If $\tilde{c}_1$ is known then we will compute $\tilde{c}_a$, and if $\tilde{d}_1$ is known then we will compute $\tilde{d}_a$. In both cases, we can determine $c_a = \tilde{c}_a + \tilde{c}_a^q = \tilde{d}_a + \tilde{d}_a^q$.

Now, by Lemma 2.4.9 one can determine $\{\tilde{c}_1, \tilde{d}_1\}$ from c_1 by finding the roots of the polynomial $\tilde{f}_g(x) = x^2 + c_1 x + c_1^t$ in $\mathbb{F}_{q^2}$. By the argument in the previous paragraph, we may assume without loss of generality that $\tilde{c}_1$ is known. Note that the minimal polynomial of g over $\mathbb{F}_{q^2}$ is $f_{g,2}(x) = x^2 + \tilde{c}_1 x + 1$, and for all integers u and v we have the following recursive relation (see [83] or Section 2.3.1):

$$\tilde{c}_{u+v} = \tilde{c}_u \tilde{c}_v - \tilde{c}_{u-v}.$$

In particular, since the characteristic of the field is 2, we have $\tilde{c}_{2u} = \tilde{c}_u^2$ and $\tilde{c}_{2u+1} = \tilde{c}_{u+1}\tilde{c}_u + \tilde{c}_1$. Thus, if we define $s_u = [\tilde{c}_u, \tilde{c}_{u+1}]$, then $s_1 = [\tilde{c}_1, \tilde{c}_1^2]$, $s_{2u} = [\tilde{c}_u^2, \tilde{c}_{u+1}\tilde{c}_u + \tilde{c}_1]$, and $s_{2u+1} = [\tilde{c}_{u+1}\tilde{c}_u + \tilde{c}_1, \tilde{c}_{u+1}^2]$. This leads to the double-and-add algorithm described in Algorithm 2.2. We note that Algorithm 2.2 can be used to compute c_{ab} given c_a and b (cf. Remark 2.4.8).

Algorithm 2.2 A single-exponentiation algorithm based on LUC-exponentiation
Input: c_1 and a
Output: c_a

1: Write $a = \sum_{i=0}^{\ell-1} a_i 2^i$ where $a_i \in \{0, 1\}$ and $a_{\ell-1} = 1$
2: $\tilde{c}_1 \leftarrow$ a root of the polynomial $x^2 + c_1 x + c_1^t$, $\tilde{c}_1 \in \mathbb{F}_{q^2}$
3: $s_u = [\tilde{c}_u, \tilde{c}_{u+1}] \leftarrow [\tilde{c}_1, \tilde{c}_1^2]$
4: **for** i from $\ell - 2$ down to 0 **do**
5: $\quad \tilde{c}_{2u+1} \leftarrow \tilde{c}_{u+1}\tilde{c}_u + \tilde{c}_1$
6: $\quad$ **if** $a_i = 1$ **then**
7: $\quad\quad \tilde{c}_{2u+2} \leftarrow \tilde{c}_{u+1}^2$
8: $\quad\quad s_u \leftarrow [\tilde{c}_{2u+1}, \tilde{c}_{2u+2}]$
9: $\quad$ **else**
10: $\quad\quad \tilde{c}_{2u} \leftarrow \tilde{c}_u^2$
11: $\quad\quad s_u \leftarrow [\tilde{c}_{2u}, \tilde{c}_{2u+1}]$
12: $\quad$ **end if**
13: **end for**
14: Return $(\tilde{c}_u + \tilde{c}_u^q)$

We may ignore the costs $(1F_1 + 1S_2)$ and $1F_2$ in the precomputation steps and in step 14 of Algorithm 2.2 as they are dominated by $SR_{2,1}$ and $(1M_2 + 1S_2)(\ell - 1)$, respectively. Then the cost of Algorithm 2.2 can be approximated as:

Precomputation (steps 2 and 3): $1SR_{2,1}$.

Main loop (steps 4–14): $(1M_2 + 1S_2)(\ell - 1)$.

Decompressing and direct exponentiation in μ_n (Algorithm DDE)

Given c_a and an ℓ-bit integer b, in order to compute c_{ab} we will decompress c_a to g^a (or to one of its conjugates over $\mathbb{F}_q$). Then we will compute g^{ab} (up to conjugation over $\mathbb{F}_q$). Finally, summing the four conjugates of g^{ab} over $\mathbb{F}_q$ gives c_{ab}.

In order to decompress c_a we first construct the polynomial $\tilde{f}_{g^a}(x) = x^2 + c_a x + c_a^t$ (see Lemma 2.4.9) over $\mathbb{F}_q$ and find a root in $\mathbb{F}_{q^2}$; without loss of generality, suppose that this root is $\tilde{c}_a$. Next we construct the minimal polynomial of g^a over $\mathbb{F}_{q^2}$, i.e., $f_{g^a,2}(x) = x^2 + \tilde{c}_a x + 1$ and find a root of $f_{g^a,2}(x)$ in $\mathbb{F}_{q^4}$. Hence we obtain g^a or one of its conjugates over $\mathbb{F}_q$. The decompression can be achieved at a cost of $1SR_{2,1} + 1SR_{2,2}$ (we ignore the cost $1F_1$ that is dominated by $SR_{2,1}$ and $SR_{2,2}$).

Now, to exponentiate $g^a \in \mu_n$ (or one of its conjugates over $\mathbb{F}_q$) to the power b, one first determines the *width-w NAF* representation of b, i.e., $b = \sum_{i=0}^{\ell'} b_i 2^i$ where $\ell' \in \{\ell - 1, \ell\}$, $b_{\ell'} \neq 0$, each nonzero b_i is odd, $|b_i| < 2^{w-1}$, and at most one of any w consecutive digits is nonzero. The width-w NAF representation of b contains on average $\ell/(w+1)$ nonzero digits (see [69] for properties of width-w NAF representations). After precomputing and storing elements $g_i = g^{ai}$ for $i \in \{\pm 1, \pm 3, \pm 5, \ldots, \pm 2^{w-1} - 1\}$, one can compute g^{ab} at an average cost of ℓ squarings and $\ell/(w+1)$ multiplications in $\mathbb{F}_{q^4}$. Finally, we note that Karatsuba's technique can be used to multiply two elements in $\mathbb{F}_{q^4}$ at a cost of $9M_1$, and squaring in μ_n can be performed at a cost of $4S_1$. Note that by choosing a suitable polynomial for the extension $\mathbb{F}_{q^4}/\mathbb{F}_q$, we may ignore the cost of polynomial reductions in the extension field arithmetic. We may also ignore the cost for computing g_i's in the precomputation step as it is dominated by $SR_{2,1}$ and $SR_{2,2}$ and the cost for computing the sum of the four conjugates of g^{ab}.

Hence, the expected cost of computing c_{ab} can be approximated as $1SR_{2,1} + 1SR_{2,2} + (4S_1 + \frac{9}{(w+1)}M_1)\ell$.

Direct exponentiation in μ_n without decompressing (Algorithm BPV-I)

This algorithm is based on the idea of Brouwer, Pellikaan and Verheul (see [13] or Section 2.3.2). Suppose c_a and an ℓ-bit integer b are given. By Lemma 2.4.9, we can determine the minimal polynomial of g^a or g^{aq} over $\mathbb{F}_{q^2}$ at a cost of $1SR_{2,1}$ (we ignore the cost $1F_1$ that is dominated by $SR_{2,1}$). Without loss of generality let's assume that we know $f_{g^a,2}(x) = x^2 + \tilde{c}_a x + 1$. That is, we have a copy of $\mathbb{F}_{q^4} = \mathbb{F}_{q^2}[x]/(f_{g^a,2}(x))$ and

next we compute x^b modulo $f_{g^a,2}(x)$ using the conventional repeated square-and-multiply algorithm. Since

$$(\tau_1 x + \tau_0)^2 = (\tau_1^2 \tilde{c}_a)x + (\tau_0^2 + \tau_1^2),$$

the squaring step can be achieved at a cost of $1M_2 + 2S_2$. And, since

$$(\tau_1 x + \tau_0)x = (\tau_1 \tilde{c}_a)x + (\tau_1 + \tau_2),$$

the multiplication step can be achieved at a cost of $1M_2$. Therefore, computing $x^b = w_1 x + w_0$ with $w_i \in \mathbb{F}_{q^2}$ costs on average $((1M_2 + 2S_2) + \frac{1}{2}M_2)(\ell - 1)$. Finally, we compute $c_{ab} = \mathrm{Tr}(x^b) = \mathrm{Tr}(\mathrm{Tr}_2(w_1 x) + \mathrm{Tr}_2(w_0)) = \mathrm{Tr}(w_1 \tilde{c}_a) = w_1 \tilde{c}_a + (w_1 \tilde{c}_a)^q$ at a cost of $1F_2 + 1M_2$. Hence, the approximate expected cost of the algorithm is $1SR_{2,1} + ((1M_2 + 2S_2) + \frac{1}{2}M_2)(\ell - 1)$ (we ignore the cost $(1F_2 + 1M_2)$ in the last step that is dominated by the cost of the main loop).

Direct exponentiation in μ_n without decompressing (Algorithm BPV-II)

The idea of the algorithm is similar to Algorithm BPV-I, except that we work with a minimal polynomial over $\mathbb{F}_q$ instead of $\mathbb{F}_{q^2}$. Given c_a and b, we first determine $f_{g^a}(x) = x^4 + c_a x^3 + c_a^t x^2 + c_a x + 1$ at a cost of $1F_1$. Now, we have a copy of $\mathbb{F}_{q^4} = \mathbb{F}_q[x]/(f_{g^a}(x))$ and next we compute x^b modulo $f_{g^a}(x)$ using the conventional repeated square-and-multiply algorithm. Since

$$x^6 \equiv (c_a^3 + c_a)x^3 + (c_a^{2t} + c_a^{t+2} + c_a^2 + 1)x^2 + (c_a^{t+1} + c_a^3 + c_a)x + (c_a^t + c_a^2) \pmod{f_{g^a}(x)},$$

and

$$(\tau_3 x^3 + \tau_2 x^2 + \tau_1 x + \tau_0)^2 = \tau_3^2 x^6 + \tau_2^2 x^4 + \tau_1^2 x^2 + \tau_0^2,$$

the squaring step can be achieved at a cost of $6M_1 + 4S_1$. And, since

$$(\tau_3 x^3 + \tau_2 x^2 + \tau_1 x + \tau_0)x = (\tau_3 c_a + \tau_2)x^3 + (\tau_3 c_a^t + \tau_1)x^2 + (\tau_3 c_a + \tau_0)x + \tau_3,$$

the multiplication step can be achieved at a cost of $2M_1$. Therefore, computing $x^b = w_3 x^3 + w_2 x^2 + w_1 x + w_0$ with $w_i \in \mathbb{F}_q$ costs $((6M_1 + 4S_1) + 1M_1)(\ell - 1)$. Finally, using $x^4 = c_a x^3 + c_a^t x^2 + c_a x + 1$ in $\mathbb{F}_{q^4} = \mathbb{F}_q[x]/(f_{g^a}(x))$, we can compute $c_{ab} = \mathrm{Tr}(x^b) = \mathrm{Tr}(w_3 x^3 + w_2 x^2 + w_1 x + w_0) = w_3 c_{3a} + w_2 c_{2a} + w_1 c_a = w_3 c_a (c_a^2 + c_a^t + 1) + w_2 c_a^2 + w_1 c_a$ at a cost of $1F_1 + 4M_1 + 1S_1$. Hence, the approximate expected cost of the algorithm is $1F_1 + (7M_1 + 4S_1)(\ell - 1)$ (we ignore the cost $(1F_1 + 4M_1 + 1S_1)$ in the last step that is dominated by the cost of the main loop).

2.5 Multiplicative groups with embedding degree 6

In this section we concentrate on multiplicative groups μ_n with embedding degree $k = 6$ over $\mathbb{F}_q$. In other words, we fix parameters (q, n) such that q is a prime power, n is a prime, $\gcd(q, n) = 1$, $q^6 \equiv 1 \pmod{n}$, and $q^i \not\equiv 1 \pmod{n}$ for $1 \leq i < 6$. Finally, we let h be a positive integer and define $t_h = q + 1 - h \cdot n$ to be the *trace* of μ_n over $\mathbb{F}_q$ with respect to the *cofactor* h. Throughout the rest of this section we will assume that the cofactor h is fixed, and we simply denote the trace of μ_n by t instead of t_h.

In the following lemma we show that $g \in \mu_n$ together with its conjugates can be uniquely represented by the triple $(\mathrm{Tr}(g), \mathrm{Tr}(g^t), \mathrm{Tr}(g^2))$ of $\mathbb{F}_q$-elements. In fact, it is easy to show that $\mathrm{Tr}(g^2)$ can be written in terms of $\mathrm{Tr}(g)$ and $\mathrm{Tr}(g^t)$. This already gives us compression by a factor of 3 . Furthermore, as first proven by Shirase et al. [78], Corollary 2.5.5 shows that in characteristic-three finite fields it is possible to write all the coefficients of the minimal polynomial of g over $\mathbb{F}_q$ in terms of $\mathrm{Tr}(g)$ alone and hence achieve compression by a factor of 6.

Lemma 2.5.1. *Let μ_n be the multiplicative subgroup of $\mathbb{F}_{q^6}^*$ of order n with embedding degree 6, trace t, and cofactor h. Let $g \in \mu_n$ and let g_i, for $i = 0, 1, \ldots, 5$, be the conjugates of g over $\mathbb{F}_q$. (Recall the convention that $g_i = g_{i \bmod 6}$.) Then*
(i) $g_i g_{i+1} = g_i^t$ for $i = 0, 1, \ldots, 5$.
(ii) $g_i g_{i+2} = g_{i+1}$ for $i = 0, 1, \ldots, 5$.
(iii) $g_i g_{i+3} = 1$ for $i = 0, 1, 2$.
(iv) $f_g(x) = x^6 - \mathrm{Tr}(g)x^5 + (\mathrm{Tr}(g^t) + \mathrm{Tr}(g) + 3)x^4 - (\mathrm{Tr}(g^2) + 2\mathrm{Tr}(g) + 2)x^3 + (\mathrm{Tr}(g^t) + \mathrm{Tr}(g) + 3)x^2 - \mathrm{Tr}(g)x + 1$.

Proof. (i) $g_i g_{i+1} = g_i^{q+1} = g_i^t$ since g_i is of order n and $q + 1 - t \equiv 0 \pmod{n}$.
(ii) $g_i g_{i+2} = g_i^{q^2+1} = g_{i+1}$ since g_i is of order n and $q^2 + 1 \equiv q \pmod{n}$.
(iii) $g_i g_{i+3} = g_i^{q^3+1} = 1$ since g_i is of order n and $q^3 + 1 \equiv 0 \pmod{n}$.
(iv) By (iii) we can write

$$\begin{aligned} f_g(x) &= \prod_{i=0}^{5}(x - g_0) \\ &= x^6 - \left(\sum_{i=0}^{5} g_i\right) x^5 + \left(\sum_{0 \leq i < j \leq 5} g_i g_j\right) x^4 - \left(\sum_{0 \leq i < j < k \leq 5} g_i g_j g_k\right) x^3 \\ &\quad + \left(\sum_{0 \leq i < j \leq 5} g_i g_j\right) x^2 - \left(\sum_{i=0}^{5} g_i\right) x + 1. \end{aligned}$$

Moreover, by (i), (ii), and (iii) we have

$$\begin{aligned}\sum_{0\le i<j\le 5} g_i g_j &= \sum_{i=0}^{5} g_i g_{i+1} + \sum_{i=0}^{5} g_i g_{i+2} + \sum_{i=0}^{2} g_i g_{i+3} \\ &= \sum_{i=0}^{5} g_i^t + \sum_{i=0}^{5} g_{i+1} + \sum_{i=0}^{2} 1 \\ &= \mathrm{Tr}(g^t) + \mathrm{Tr}(g) + 3\end{aligned}$$

and

$$\begin{aligned}\sum_{0\le i<j<k\le 5} g_i g_j g_k &= \sum_{i=0}^{5} g_i g_{i+1} g_{i+2} + \sum_{i=0}^{2} \sum_{\substack{j=0 \\ j\not\equiv i,i+3 \pmod 6}}^{5} g_i g_{i+3} g_j + \sum_{i=0}^{1} g_i g_{i+2} g_{i+4} \\ &= \sum_{i=0}^{5} g_{i+1}^2 + 2\sum_{j=0}^{5} g_j + 2\end{aligned}$$

which completes the proof. □

Suppose we fix a generator $g \in \mathbb{F}_{q^6}$ of μ_n, that is $\mu_n = \langle g \rangle$. In order to simplify the notation we define $c_u = \mathrm{Tr}(g^u)$ for any integer u. Note that $c_0 = 0$, $c_1 = \mathrm{Tr}(g)$, $c_u = c_{u \bmod n}$ and $c_{qu} = c_u^q = c_u$.

Lemma 2.5.2. *Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}_{q^6}^*$ of order n with embedding degree 6 and trace t. Then for all integers u and v we have*
(i) $c_u = c_{-u}$.
(ii) $c_u c_v = c_{u+v} + c_{u-v} + c_{u+v(t-1)} + c_{v+u(t-1)} + c_{u+v(t-2)} + c_{v+u(t-2)}$.

Proof. (i) This follows from Lemma 2.5.1 (iii).
(ii) First note that

$$\begin{aligned}c_u c_v &= (g_0^u + g_1^u + g_2^u + g_3^u + g_4^u + g_5^u)(g_0^v + g_1^v + g_2^v + g_3^v + g_4^v + g_5^v) \\ &= \sum_{i=0}^{5} g_i^{u+v} + \sum_{i=0}^{5} g_i^u g_{i+1}^v + \sum_{i=0}^{5} g_i^u g_{i+2}^v \\ &\quad + \sum_{i=0}^{5} g_i^u g_{i+3}^v + \sum_{i=0}^{5} g_i^u g_{i+4}^v + \sum_{i=0}^{5} g_i^u g_{i+5}^v.\end{aligned}$$

Observing the equations below with the help of Lemma 2.5.1(i), (ii) and (iii)

$$\begin{aligned}\sum_{i=0}^{5} g_i^u g_{i+1}^v &= \sum_{i=0}^{5} g_i^{u-v}(g_i^v g_{i+1}^v) = \sum_{i=0}^{5} g_i^{u+v(t-1)},\\ \sum_{i=0}^{5} g_i^u g_{i+2}^v &= \sum_{i=0}^{5} g_i^{u-v}(g_i^v g_{i+2}^v) = \sum_{i=0}^{5} g_i^{u-v} g_{i+1}^v = \sum_{i=0}^{5} g_i^{u+v(t-2)},\end{aligned}$$

$$\begin{aligned}\sum_{i=0}^{5} g_i^u g_{i+3}^v &= \sum_{i=0}^{5} g_i^{u-v}(g_i^v g_{i+3}^v) = \sum_{i=0}^{5} g_i^{u-v},\\ \sum_{i=0}^{5} g_i^u g_{i+4}^v &= \sum_{i=0}^{5} g_i^v g_{i+2}^u = \sum_{i=0}^{5} g_i^{v+u(t-2)},\\ \sum_{i=0}^{5} g_i^u g_{i+5}^v &= \sum_{i=0}^{5} g_i^v g_{i+1}^u = \sum_{i=0}^{5} g_i^{v+u(t-1)}\end{aligned}$$

leads us to the result. □

Theorem 2.5.3. *Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}_{q^6}^*$ of order n with embedding degree 6 and trace t. Then for all integers u and v we have*

$$c_{u+v} = (c_u + c_{u-4v})c_v - (c_{u-v} + c_{u-3v})(c_{tv} + c_v + 3) + c_{u-2v}(2c_v + c_{2v} + 2) - c_{u-5v}.$$

Proof. First we start with

$$\begin{aligned}c_u c_v &= (g_0^u + g_1^u + g_2^u + g_3^u + g_4^u + g_5^u)(g_0^v + g_1^v + g_2^v + g_3^v + g_4^v + g_5^v)\\ &= \sum_{i=0}^{5} g_i^{u+v} + \sum_{i=0}^{5}\sum_{\substack{j=0\\ j\neq i}}^{5} g_i^u g_j^v\\ &= c_{u+v} + \sum_{i=0}^{5}\sum_{\substack{j=0\\ j\neq i}}^{5} g_i^{u-v}(g_i^v g_j^v)\\ &= c_{u+v} + \sum_{i=0}^{5} g_i^{u-v}(g_i^{tv} + g_{i+5}^{tv} + g_{i+1}^v + g_{i+5}^v + 1)\\ &= c_{u+v} + c_{u-v} + c_{u-v}c_{tv} + c_{u-v}c_v\\ &\quad - \left(\sum_{i=0}^{5} g_i^{u-v}(g_{i+1}^{tv} + g_{i+2}^{tv} + g_{i+3}^{tv} + g_{i+4}^{tv})\right)\\ &\quad - \left(\sum_{i=0}^{5} g_i^{u-v}(g_i^v + g_{i+2}^v + g_{i+3}^v + g_{i+4}^v)\right).\end{aligned}$$

Now, since

$$\begin{aligned}
\sum_{i=0}^{5} g_i^{u-v} g_{i+1}^{tv} &= \sum_{i=0}^{5} g_i^{u-v+qtv}, \text{ by Lemma 2.5.1(i)} \\
&= c_{q^4(u-v)+q^5tv}, \text{ since } c_{qu} = c_u \\
&= c_{qu-2v}, \text{ by Lemma 2.5.2(i) and } q^4 \equiv -q \pmod{n}, \\
\sum_{i=0}^{5} g_i^{u-v} g_{i+2}^{tv} &= \sum_{i=0}^{5} g_i^{u-v+q^2tv}, \text{ by Lemma 2.5.1(i)} \\
&= c_{(u-3v)+qv} \text{ since } q^2t \equiv q-2 \pmod{n}, \\
\sum_{i=0}^{5} g_i^{u-v} g_{i+3}^{tv} &= \sum_{i=0}^{5} g_i^{u-v+q^3tv}, \text{ by Lemma 2.5.1(i)} \\
&= c_{(u-2v)-qv} \text{ since } q^3 \equiv -1 \pmod{n}, \\
\sum_{i=0}^{5} g_i^{u-v} g_{i+4}^{tv} &= \sum_{i=0}^{5} g_i^{u-v+q^4tv}, \text{ by Lemma 2.5.1(i)} \\
&= c_{u-2qv}, \text{ by Lemma 2.5.2(i) and } q^4t \equiv -2q+1 \pmod{n},
\end{aligned}$$

$$\begin{aligned}
\sum_{i=0}^{5} g_i^{u-v} g_{i+2}^{v} &= \sum_{i=0}^{5} g_i^{u-v+q^2v}, \text{ by Lemma 2.5.1(i)} \\
&= c_{(u-2v)+qv} \text{ since } q^2 \equiv q-1 \pmod{n}, \\
\sum_{i=0}^{5} g_i^{u-v} g_{i+3}^{v} &= \sum_{i=0}^{5} g_i^{u-v+q^3v}, \text{ by Lemma 2.5.1(i)} \\
&= c_{(u-2v)} \text{ since } q^3 \equiv -1 \pmod{n},
\end{aligned}$$

and

$$\begin{aligned}
\sum_{i=0}^{5} g_i^{u-v} g_{i+4}^{v} &= \sum_{i=0}^{5} g_i^{u-v+q^4v}, \text{ by Lemma 2.5.1(i)} \\
&= c_{(u-v)-qv} \text{ since } q^4 \equiv -q \pmod{n},
\end{aligned}$$

we have

$$\begin{aligned}
c_u c_v &= c_{u+v} + c_{u-v} + c_{u-v}c_{tv} + c_{u-v}c_v \\
&\quad -(c_{qu-2v} + c_{(u-3v)+qv} + c_{(u-2v)-qv} + c_{u-2qv}) \\
&\quad -(c_u + c_{(u-2v)+qv} + c_{u-2v} + c_{(u-v)-qv}).
\end{aligned} \tag{2.5.1}$$

In order to eliminate some of the terms in equation 2.5.1 let us use Lemma 2.5.2(ii) with $u' = u - 2v$, and $v' = qv$. By noting that

$$\begin{aligned}
c_{v'} &= c_v, \text{ since } c_{qu} = c_u, \\
c_{u'+v'} &= c_{(u-2v)+qv}, \\
c_{u'-v'} &= c_{(u-2v)-qv}, \\
c_{u'+v'(t-1)} &= c_{(u-3v)+qv}, \text{ since } q(t-1) \equiv q^2 \equiv q-1 \pmod{n}, \\
c_{u'+v'(t-2)} &= c_{(u-3v)}, \\
c_{v'+u'(t-1)} &= c_{u-v}, \text{ since } t-1 \equiv q \pmod{n} \text{ and } c_{qu} = c_u, \\
c_{v'+u'(t-2)} &= c_{q(q(u-v)-(u-2v))}, \text{ since } c_{qu} = c_u \\
&= c_{(u-v)-qv}, \text{by Lemma 2.5.2(i)},
\end{aligned}$$

we can rewrite (2.5.1) as

$$\begin{aligned}
c_u c_v &= c_{u+v} + 2c_{u-v} + c_{u-v}c_{tv} + c_{u-v}c_v \\
&\quad -(c_{qu-2v} + c_{u-2qv} + c_u + c_{u-2v}) \\
&\quad -(c_{u-2v}c_v - c_{u-3v}).
\end{aligned} \tag{2.5.2}$$

By replacing v by $-v$ in (2.5.2) we obtain another equation and summing this with (2.5.2) gives

$$\begin{aligned}
c_{u+3v} &= 2c_u c_v - 3(c_{u+v} + c_{u-v}) - (c_{tv} + c_v)(c_{u+v} + c_{u-v}) + 2c_u \\
&\quad +c_v(c_{u+2v} + c_{u-2v}) + (c_{u+2v} + c_{u-2v}) - c_{u-3v} + c_{qu-2v} \\
&\quad +c_{u-2qv} + c_{qu+2v} + c_{u+2qv}.
\end{aligned} \tag{2.5.3}$$

Also, using Lemma 2.5.2(ii) with $u' = u$ and $v' = 2v$, and noting that

$$\begin{aligned}
c_{u'+v'(t-1)} &= c_{u+2qv}, \text{ since } t-1 \equiv q \pmod{n}, \\
c_{u'+v'(t-2)} &= c_{q(u+2(q-1)v)}, \text{ since } c_{qu} = c_u \text{ and } t-2 \equiv q-1 \pmod{n}, \\
&= c_{qu-2v}, \text{ since } q^2 \equiv q-1 \pmod{n}, \\
c_{v'+u'(t-1)} &= c_{qu+2v}, \text{ since } t-1 \equiv q \pmod{n}, \\
c_{v'+u'(t-2)} &= c_{q(2v+(q-1)u)}, \text{ since } c_{qu} = c_u \text{ and } t-2 \equiv q-1 \pmod{n}, \\
&= c_{-u+2qv} = c_{u-2qv}, \text{ since } q^2 \equiv q-1 \pmod{n} \text{ and } c_u = c_{-u},
\end{aligned}$$

we can rewrite (2.5.3) as

$$\begin{aligned}
c_{u+3v} &= 2c_u c_v - 3(c_{u+v} + c_{u-v}) - (c_{tv} + c_v)(c_{u+v} + c_{u-v}) + 2c_u \\
&\quad +c_v(c_{u+2v} + c_{u-2v}) - c_{u-3v} + c_u c_{2v}.
\end{aligned} \tag{2.5.4}$$

Finally, using (2.5.4) with $u' = u - 2v$ and $v' = v$ results in

$$\begin{aligned} c_{u+v} &= c_u c_v - (c_{u-v} + c_{u-3v})(c_{tv} + c_v + 3) + c_{u-2v}(2c_v + c_{2v} + 2) \\ &\quad + c_{u-4v}c_v - c_{u-5v}, \end{aligned}$$

as required. □

2.5.1 Characteristic-three finite fields

This section follows along the same lines as Section 2.4.1. Let r be a positive integer, and let $q = 3^{2r+1}$, $t = \pm 3^{r+1}, T = |t|$. The values of r for which $q + 1 - t = hn$ and n is prime lead to a multiplicative subgroup μ_n of $\mathbb{F}_{q^6}^*$ of prime order n with embedding degree 6. Throughout this section we fix h, n, q, t, T and $\mu_n = \langle g \rangle$ in this way, and also write $c_u = \mathrm{Tr}(g^u)$.

The following recursive relations follow from Theorem 2.5.3, using Lemma 2.5.2(ii) with $u = v$, and by noting that the characteristic of $\mathbb{F}_q$ is 3 and also that $c_{vt} = c_{vT} = c_v^T$ (see Lemma 2.5.2(i)).

Corollary 2.5.4. *Let μ_n be the multiplicative subgroup of $\mathbb{F}_{q^6}^*$ with embedding degree 6 and trace t. Then for all integers u and v we have*
(i) $c_{2u} = c_u^2 + c_u + c_u^T$.
(ii) $c_{u+v} = c_u c_v - (c_{u-v} + c_{u-3v})(c_v^T + c_v) + c_{u-2v}(c_v^2 + c_v^T + 2) + c_{u-4v}c_v - c_{u-5v}$.

Corollary 2.5.5. *Let μ_n be the multiplicative subgroup of $\mathbb{F}_{q^6}^*$ with embedding degree 6 and trace t. Let $f_{g^u}(x)$ be the minimal polynomial of $g^u \in \mu_n$ over $\mathbb{F}_q$. Then*

$$f_{g^u}(x) = x^6 - c_u x^5 + (c_u^T + c_u)x^4 - (c_u^2 + c_u^T + 2)x^3 + (c_u^T + c_u)x^2 - c_u x + 1.$$

Proof. The proof follows from Lemma 2.5.1(iv), Corollary 2.5.4(i) and from the fact that $c_{vt} = c_{vT} = c_v^T$. □

Remark 2.5.6. Throughout the remainder of this section we will assume without loss of generality that the trace t is positive. If t is negative then one can replace the expressions of the form $c_u^{p(t)}$, where p is some polynomial, by $c_u^{p(T)}$ without changing the validity of the results in this section.

2.5.2 An exponentiation algorithm in μ_n

Corollary 2.5.5 shows that the element g^u can be represented uniquely (up to conjugation) by its trace c_u. Similarly as in Section 2.5.2, our aim is to develop an efficient method to compute c_a given c_1 and a. We define $s_1 = [c_{-1}, c_0, c_1, c_2, c_3, c_4]$ to

be the initial state. For a given state $s_u = [c_{u-2}, c_{u-1}, c_u, c_{u+1}, c_{u+2}, c_{u+3}]$ with $u \geq 1$, if we can efficiently compute the states $s_{2u} = [c_{2u-2}, c_{2u-1}, c_{2u}, c_{2u+1}, c_{2u+2}, c_{2u+3}]$ and $s_{2u+1} = [c_{2u-1}, c_{2u}, c_{2u+1}, c_{2u+2}, c_{2u+3}, c_{2u+4}]$ then we immediately have an efficient double-and-add algorithm for computing c_a given c_1 and a.

Theorem 2.5.7. *Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}_{q^6}^*$ with embedding degree 6 and trace t. Let $c_u = \mathrm{Tr}(g^u)$, $C_{1,t} = c_1 + c_1^t$, $C_{1,t,2} = c_1^2 + c_1^t + 2$, $M = (c_1^t(c_1^3 + c_1^{t+1} + 2(c_1^t + 1)) + 2(c_1^3 + c_1 + 1))^{-1}$, and*

$$A = \begin{pmatrix} 1 & C_{1,t} & C_{1,t} & 1 & 0 & 0 \\ 0 & 1 & C_{1,t} & C_{1,t} & 1 & 0 \\ 0 & 0 & 1 & C_{1,t} & C_{1,t} & 1 \\ c_1 & C_{1,t,2} & c_1 & 0 & 0 & 0 \\ 0 & c_1 & C_{1,t,2} & c_1 & 0 & 0 \\ 0 & 0 & c_1 & C_{1,t,2} & c_1 & 0 \end{pmatrix}, \; X = \begin{pmatrix} c_{2u-3} \\ c_{2u-1} \\ c_{2u+1} \\ c_{2u+3} \\ c_{2u+5} \\ c_{2u+7} \end{pmatrix},$$

$$Y = \begin{pmatrix} (c_{2u+2} + c_{2u-2})c_1 + c_{2u}C_{1,t,2} \\ (c_{2u+4} + c_{2u})c_1 + c_{2u+2}C_{1,t,2} \\ (c_{2u+6} + c_{2u+2})c_1 + c_{2u+4}C_{1,t,2} \\ (c_{2u} + c_{2u-2})C_{1,t} + c_{2u+2} + c_{2u-4} \\ (c_{2u+2} + c_{2u})C_{1,t} + c_{2u+4} + c_{2u-2} \\ (c_{2u+4} + c_{2u+2})C_{1,t} + c_{2u+6} + c_{2u} \end{pmatrix}.$$

Then
(i) A is invertible and $AX = Y$.
(ii) If c_1 is given then A and A^{-1} can be efficiently computed.
(iii) $c_{2u-1} = M(c_1^3 + 2c_1^2 + 2c_1 + c_1^{t+1}, c_1^2, 0, 2(c_1^2 + c_1^t) + c_1 + 1, 2(c_1^2 + c_1^{t+1}) + c_1, 2c_1)Y$.
(iv) $c_{2u+1} = M(2c_1^2, 2c_1^2, 0, c_1, 2c_1^t + c_1^{t+1} + c_1 + 1, c_1)Y$.
(v) $c_{2u+3} = M(c_1^2, c_1^3 + 2c_1^2 + 2c_1 + c_1^{t+1}, 0, 2c_1, 2(c_1^2 + c_1^{t+1}) + c_1, 2(c_1^2 + c_1^t) + c_1 + 1)Y$.

Proof. (i) Noting that the characteristic of $\mathbb{F}_q$ is 3 and using Corollary 2.5.4(i), we can show that determinant of A is equal to $(c_1 - 1)^{2t+2}(c_2 - 1)$. Hence A is invertible if and only if $c_1 \neq 1$ and $c_2 \neq 1$. In fact we can show that $c_1 \neq 1$ and $c_2 \neq 1$ as follows. If $c_1 = 1$ then $f_g(x) = x^6 - x^5 + 2x^4 - x^3 + 2x^2 - x + 1 = (x^2 + x + 1)(x^2 - x + 1)^2$, and if $c_2 = 1$ then $f_g(x) = (x^2 + 1)(x^4 - c_1x^3 + (c_1 + c_1^t - 1)x^2 - c_1x + 1)$. Both cases contradict the irreducibility of the minimal polynomial $f_g(x)$ over $\mathbb{F}_q$. This proves that A is invertible. Now, combining the six equations obtained from Corollary 2.5.4 with the following (u, v) values: $(2u-3, -1), (2u-2, -1), (2u-1, -1), (2u, -1), (2u+1, -1), (2u+2, -1)$ and using Corollary 2.5.4(i) proves that $AX = Y$.
(ii) The proof follows from part (i) and from the definition of A.

(iii) A computation in Maple shows that the second row of A^{-1} is $A^{-1}[2] = \frac{1}{M}(c_1^3 + 2c_1^2 + 2c_1 + c_1^{t+1}, c_1^2, 0, 2(c_1^2 + c_1^t) + c_1 + 1, 2(c_1^2 + c_1^{t+1}) + c_1), 2c_1)$. The rest follows as $c_{2u-1} = A^{-1}[2]Y$ from part (i).
(iv) We compute $A^{-1}[3] = \frac{1}{M}(2c_1^2, 2c_1^2, 0, c_1, 2c_1^t + c_1^{t+1} + c_1 + 1, c_1)$ and conclude from $c_{2u+1} = A^{-1}[3]Y$.
(v) We compute $A^{-1}[4] = \frac{1}{M}(c_1^2, c_1^3 + 2c_1^2 + 2c_1 + c_1^{t+1}, 0, 2c_1, 2(c_1^2 + c_1^{t+1}) + c_1), 2(c_1^2 + c_1^t) + c_1 + 1)$ and conclude from $c_{2u+3} = A^{-1}[4]Y$. □

The formulas for $c_{2u-1}, c_{2u+1}, c_{2u+3}$ in Theorem 2.5.7 yield Algorithm 2.3 for exponentiation in μ_n.

Remark 2.5.8. Algorithm 2.3 can be used to compute c_{ab} given c_a and b as follows. We replace $c'_1 = c_a$ and the initial state becomes $s'_1 = [c'_{-1}, c'_0, c'_1, c'_2, c'_3, c'_4] = [c_a, 0, c_a, C_{a,t,2}, c_a^3, C_{a,t,2}^2 + C_{a,t,2} + C_{a,t,2}^t]$. With input c'_1 and b, Algorithm 2.3 outputs $c'_b = c_{ab}$.

We may ignore the costs $(9A_1 + 4a_1 + 1C_1 + 1F_1 + 2m_1)$ in the precomputation steps of Algorithm 2.3 and $(3a_1 + 10m_1)(\ell - 1)$ in the main loop of Algorithm 2.3 as they are dominated by $(1I_1 + 2M_1 + 2S_1)$ and $(53A_1 + 6F_1 + 23M_1 + 6S_1)(\ell - 1)$, respectively. Then the cost of Algorithm 2.3 can be approximated as:

Precomputation (steps 2–6): $1I_1 + 2M_1 + 2S_1$.

Main loop (steps 7–18): $(53A_1 + 6F_1 + 23M_1 + 6S_1)(\ell - 1)$.

We note that Algorithm 2.3 has a limited degree of built-in resistance to side-channel analysis attacks as the same types of operations are executed whether the bit a_i of the exponent is 1 or 0.

2.5.3 Other algorithms for exponentiation with compressed elements

Algorithm 2.3 works directly with the factor-6 compressed elements. In this section, we describe five algorithms for computing c_{ab} given c_a and b. The first algorithm partially decompresses c_a to an element $\tilde{c}_a \in \mathbb{F}_{q^3}$, and then uses the LUC method for exponentiating in this representation. The second algorithm decompresses c_a to an element in $\mathbb{F}_{q^6}$, and then employs a standard window-NAF exponentiation method. The third, fourth and fifth methods use the Brouwer-Pellikaan-Verheul idea (cf. Section 2.3.2) by using minimal polynomials over $\mathbb{F}_{q^2}$, $\mathbb{F}_{q^3}$ and $\mathbb{F}_q$, respectively. The seven exponentiation algorithms including XTR_3 [78] are compared in Section 2.6.

First, we prove the following.

Algorithm 2.3 A single-exponentiation algorithm
Input: c_1 and a
Output: c_a

1: Write $a = \sum_{i=0}^{\ell-1} a_i 2^i$ where $a_i \in \{0,1\}$ and $a_{\ell-1} = 1$
2: $C_2 \leftarrow c_1^2, \quad C_t \leftarrow c_1^{t+1}$
3: $C_{1,t} \leftarrow c_1 + c_1^t, \quad C_{1,t,2} \leftarrow C_2 + c_1^t + 2$
4: $c_2 \leftarrow C_{1,t,2} + c_1 + 1, \quad c_4 \leftarrow c_2^2 + c_2 + c_2^t$
5: $s_u = [c_{u-2}, c_{u-1}, c_u, c_{u+1}, c_{u+2}, c_{u+3}] \leftarrow [c_1, 0, c_1, c_2, c_1^3, c_4]$
6: $M \leftarrow (c_1^t(c_1^3 + C_t + 2(c_1^t + 1)) + 2(c_1^3 + c_1 + 1))^{-1}$
7: **for** i from $\ell - 2$ down to 0 **do**
8: $c_{2u-4} \leftarrow c_{u-2}^2 + c_{u-2} + c_{u-2}^t, \quad c_{2u-2} \leftarrow c_{u-1}^2 + c_{u-1} + c_{u-1}^t,$
$c_{2u} \leftarrow c_u^2 + c_u + c_u^t, \quad c_{2u+2} \leftarrow c_{u+1}^2 + c_{u+1} + c_{u+1}^t,$
$c_{2u+4} \leftarrow c_{u+2}^2 + c_{u+2} + c_{u+2}^t, \quad c_{2u+6} \leftarrow c_{u+3}^2 + c_{u+3} + c_{u+3}^t$
9: $Y_1 \leftarrow (c_{2u+2} + c_{2u-2})c_1 + c_{2u}C_{1,t,2},$
$Y_2 \leftarrow (c_{2u+4} + c_{2u})c_1 + c_{2u+2}C_{1,t,2},$
$Y_4 \leftarrow (c_{2u} + c_{2u-2})C_{1,t} + c_{2u+2} + c_{2u-4},$
$Y_5 \leftarrow (c_{2u+2} + c_{2u})C_{1,t} + c_{2u+4} + c_{2u-2},$
$Y_6 \leftarrow (c_{2u+4} + c_{2u+2})C_{1,t} + c_{2u+6} + c_{2u}$
10: $c_{2u-1} \leftarrow M((c_1^3 + 2(C_2 + c_1) + C_t)Y_1 + C_2Y_2 + (2(C_2 + c_1^t) + c_1 + 1)Y_4$
$+(2(C_2 + C_t) + c_1)Y_5 + 2c_1Y_6)$
11: $c_{2u+1} \leftarrow M(2C_2(Y_1 + Y_2) + c_1(Y_4 + Y_6) + (2c_1^t + C_t + c_1 + 1)Y_5)$
12: $c_{2u+3} \leftarrow M(C_2Y_1 + (c_1^3 + 2(C_2 + c_1) + C_t)Y_2 + 2c_1Y_4$
$+(2(C_2 + C_t) + c_1)Y_5 + (2(C_2 + c_1^t) + c_1 + 1)Y_6)$
13: **if** $a_i = 1$ **then**
14: $s_u \leftarrow [c_{2u-1}, c_{2u}, c_{2u+1}, c_{2u+2}, c_{2u+3}, c_{2u+4}]$
15: **else**
16: $s_u \leftarrow [c_{2u-2}, c_{2u-1}, c_{2u}, c_{2u+1}, c_{2u+2}, c_{2u+3}]$
17: **end if**
18: **end for**
19: Return (c_u)

Lemma 2.5.9. *Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}_{q^6}^*$ with embedding degree 6 and trace t. Also, let $c_u = \mathrm{Tr}(g^u)$ and $\tilde{c}_u = \mathrm{Tr}_3(g^u)$. Then $\{\tilde{c}_u, \tilde{c}_u^q, \tilde{c}_u^{q^2}\}$ is the set of roots of the polynomial $\tilde{f}_{g^u}(x) = x^3 - c_u x^2 + (c_u + c_u^t)x - (c_u^2 + c_u^t - 2c_u + 2)$.*

Proof. Let $\tilde{d}_u = \tilde{c}_u^q$ and $\tilde{e}_u = \tilde{c}_u^{q^2}$. Since g has order n and $q + 1 \equiv t \pmod{n}$ and $q^3 \equiv -1 \pmod{n}$, we can rewrite the minimal polynomial of g^u over $\mathbb{F}_q$ as

$$\begin{aligned} f_{g^u}(x) &= \left[(x - g^u)(x - g^{q^3 u})\right]\left[(x - g^{qu})(x - g^{q^4 u})\right]\left[(x - g^{q^2 u})(x - g^{q^5 u})\right] \\ &= (x^2 - \tilde{c}_u x + 1)(x^2 - \tilde{d}_u x + 1)(x^2 - \tilde{e}_u x + 1) \\ &= x^6 - c_u x^5 + (\tilde{c}_u\tilde{d}_u + \tilde{d}_u\tilde{e}_u + \tilde{c}_u\tilde{e}_u)x^4 - (2c_u + \tilde{c}_u\tilde{d}_u\tilde{e}_u)x^3 \\ &\quad +(\tilde{c}_u\tilde{d}_u + \tilde{d}_u\tilde{e}_u + \tilde{c}_u\tilde{e}_u)x^2 - c_u x + 1. \end{aligned}$$

Comparing the coefficients of this polynomial with the coefficients of $f_{g^u}(x)$ as given in Corollary 2.5.5 we get

$$\begin{aligned} \tilde{c}_u\tilde{d}_u + \tilde{d}_u\tilde{e}_u + \tilde{c}_u\tilde{e}_u &= c_u + c_u^t \\ \tilde{c}_u\tilde{d}_u\tilde{e}_u &= c_u^2 + c_u^t - 2c_u + 2. \end{aligned}$$

The proof then follows because

$$\begin{aligned} (x - \tilde{c}_u)(x - \tilde{d}_u)(x - \tilde{e}_u) &= x^3 - (\tilde{c}_u + \tilde{d}_u + \tilde{e}_u)x^2 + (\tilde{c}_u\tilde{d}_u + \tilde{d}_u\tilde{e}_u + \tilde{c}_u\tilde{e}_u)x \\ &\quad -\tilde{c}_u\tilde{d}_u\tilde{e}_u \\ &= x^3 - c_u x^2 + (c_u + c_u^t)x - (c_u^2 + c_u^t - 2c_u + 2) \\ &= \tilde{f}_{g^u}(x). \end{aligned}$$

□

We will also need the following lemma which was proven in [78].

Lemma 2.5.10. *Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}_{q^6}^*$ with embedding degree 6 and trace t. Also, let $c_u = \mathrm{Tr}(g^u)$ and $\tilde{c}_u = \mathrm{Tr}_2(g^u)$. Then $\{\tilde{c}_u, \tilde{c}_u^q\}$ is the set of roots of the polynomial $\tilde{f}_{g^u}(x) = x^2 - c_u x + c_u^t$.*

An algorithm based on the LUC cryptosystem

We will describe a ternary exponentiation algorithm to compute c_a given c_1 and a. The idea of the algorithm is as follows. Suppose we know an element in the set $\{\tilde{c}_1, \tilde{c}_1^q, \tilde{c}_1^{q^2}\}$. If $\tilde{c}_1$ is known then we will compute $\tilde{c}_a$, if $\tilde{c}_1^q$ is known then we will compute $\tilde{c}_a^q$, and if $\tilde{c}_1^{q^2}$ is

known then we will compute $\tilde{c}_a^{q^2}$. In all three cases, we can determine $c_a = \tilde{c}_a + \tilde{c}_a^q + \tilde{c}_a^{q^2} = \tilde{c}_a^q + (\tilde{c}_a^q)^q + (\tilde{c}_a^q)^{q^2} = \tilde{c}_a^{q^2} + (\tilde{c}_a^{q^2})^q + (\tilde{c}_a^{q^2})^{q^2}$.

By Lemma 2.5.9 we can determine $\{\tilde{c}_1, \tilde{c}_1^q, \tilde{c}_1^{q^2}\}$ based on c_1 by finding the roots of the polynomial $\tilde{f}_g(x) = x^3 - c_1x^2 + (c_1 + c_1^t)x - (c_1^2 + c_1^t - 2c_1 + 2)$. From our argument in the previous paragraph, we may assume without loss of generality that $\tilde{c}_1$ is known. Note that the minimal polynomial of g over $\mathbb{F}_{q^3}$ is $f_{g,3}(x) = x^2 - \tilde{c}_1 x + 1$, and for all integers u and v we have the following recursive relation (see [83] or Section 2.3.1):

$$\tilde{c}_{u+v} = \tilde{c}_u\tilde{c}_v - \tilde{c}_{u-v}. \tag{2.5.5}$$

Moreover, it was shown in [77] that the following equations can be deduced from (2.5.5).

$$\begin{aligned}
\tilde{c}_{3u} &= \tilde{c}_u^3 && (2.5.6)\\
\tilde{c}_{3u-1} &= \frac{1}{\tilde{c}_1^2 - 1}(\tilde{c}_{u-1}^3 + \tilde{c}_1\tilde{c}_u^3) && (2.5.7)\\
\tilde{c}_{3u-1} &= \frac{1}{\tilde{c}_1^2 - 1}((\tilde{c}_1^3 + \tilde{c}_1)\tilde{c}_u^3 - \tilde{c}_{u+1}^3) && (2.5.8)\\
\tilde{c}_{3u+1} &= \frac{1}{\tilde{c}_1^2 - 1}((\tilde{c}_1^3 + \tilde{c}_1)\tilde{c}_u^3 - \tilde{c}_{u-1}^3) && (2.5.9)\\
\tilde{c}_{3u+1} &= \frac{1}{\tilde{c}_1^2 - 1}(\tilde{c}_{u+1}^3 + \tilde{c}_1\tilde{c}_u^3). && (2.5.10)
\end{aligned}$$

We now describe how to obtain $\tilde{c}_a$ given $\tilde{c}_1$ and a; to the author's knowledge this exponentiation method was first presented in [77] to compute Lucas sequences. We will assume that a is written in signed ternary notation, i.e., $a = \sum_{i=0}^{\ell-1} a_i 3^i$ where $a_i \in \{-1, 0, 1\}$ and $a_{\ell-1} = 1$ First, define two states: $s_u^{(0)} = [\tilde{c}_u, \tilde{c}_{u+1}]$ and $s_u^{(1)} = [\tilde{c}_{u-1}, \tilde{c}_u]$. The algorithm begins with the state $s_1^{(1)} = [\tilde{c}_0, \tilde{c}_1] = [2, \tilde{c}_1]$, and exactly one of the two states, $s_u^{(j)}, j = 0, 1$, will be active during the execution of the algorithm.

If $j = 0$ in the ith step of the algorithm, then $s_u^{(0)}$ is active and we will compute $\tilde{c}_{3u}$ based on (2.5.6) and compute one of $\tilde{c}_{3u-1}$ or $\tilde{c}_{3u+1}$ based on (2.5.8) or (2.5.10). In this case, if $a_i = 0$ then we set $j = 0$ (i.e., $s_u^{(0)}$ remains active) and $s_u^{(j)} \leftarrow [\tilde{c}_{3u}, \tilde{c}_{3u+1}]$; if $a_i = 1$ we set $j = 1$ (i.e., $s_u^{(1)}$ becomes active) and $s_u^{(j)} \leftarrow [\tilde{c}_{3u}, \tilde{c}_{3u+1}]$; and if $a_i = -1$ we set $j = 0$ (i.e., $s_u^{(0)}$ remains active) and $s_u^{(j)} \leftarrow [\tilde{c}_{3u}, \tilde{c}_{3u+1}]$.

If $j = 1$ in the ith step of the algorithm, then $s_u^{(1)}$ is active and we will compute $\tilde{c}_{3u}$ based on (2.5.6) and compute one of $\tilde{c}_{3u-1}$ or $\tilde{c}_{3u+1}$ based on (2.5.7) or (2.5.9). In this case, if $a_i = 0$ then we set $j = 1$ (i.e., $s_u^{(1)}$ remains active) and $s_u^{(j)} \leftarrow [\tilde{c}_{3u-1}, \tilde{c}_{3u}]$; if $a_i = 1$ we set $j = 1$ (i.e., $s_u^{(1)}$ remains active) and $s_u^{(j)} \leftarrow [\tilde{c}_{3u}, \tilde{c}_{3u+1}]$; and if $a_i = -1$ we set $j = 0$ (i.e., $s_u^{(0)}$ becomes active) and $s_u^{(j)} \leftarrow [\tilde{c}_{3u-1}, \tilde{c}_{3u}]$.

At the end of this procedure we obtain $\tilde{c}_a$ from one of the active states $s_a^{(0)} = [\tilde{c}_a, \tilde{c}_{a+1}]$ or $s_a^{(1)} = [\tilde{c}_{a-1}, \tilde{c}_a]$, as required. The exponentiation method is summarized in Algorithm 2.4. We may ignore the costs $(2A_1+2a_1+1A_3+1C_3+1F_1+1I_1+2m_1+1S_1)$ in the precomputation steps of Algorithm 2.4 and $(1m_3)(\ell-1)+(2A_3+2F_3)$ in the main loop of Algorithm 2.4 as they are dominated by $1SR_{3,1}$ and $(2C_3+2M_3)(\ell-1)$, respectively. Then the cost of Algorithm 2.4 can be approximated as:

Precomputation (steps 2–4): $1SR_{3,1}$.

Main loop (steps 5–37): $(1A_3+2C_3+2M_3)(\ell-1)$.

Remark 2.5.11. Montgomery [68] presented several methods (binary, binary-ternary, CFRC, PRAC) to compute Lucas sequences. The main idea in his methods to compute $\tilde{c}_a$ is to build a Lucas chain for a and at each step in the chain to use the recursive formula $\tilde{c}_{u+v} = \tilde{c}_u\tilde{c}_v - \tilde{c}_{u-v}$ for some suitable u and v (see Table 4 in [68]). The length of the derived Lucas chains in these algorithms exceed $1.446(\log_2 a)$ (see Theorem 8 and Table 5 in [68]) and each step in the chain requires at least $1M_3$ (see Table 4 in [68]). Therefore, these methods seem unlikely to outperform Algorithm 2.4 whose cost might be approximated as $2(\log_3 2)(\log_2 a)M_3 \approx 1.26(\log_2 a)M_3$ after the precomputation step.

Decompressing and direct exponentiation in μ_n (Algorithm DDE)

Given c_a and an integer b, in order to compute c_{ab} we will first decompress c_a to g^a or one of its conjugates over $\mathbb{F}_q$. Then we will compute g^{ab} (up to conjugation over $\mathbb{F}_q$) by working directly in $\mathbb{F}_{q^6}$. Finally, summing the six conjugates of g^{ab} gives c_{ab}.

In order to decompress g^a we first construct the polynomial $\tilde{f}_{g^a}(x) = x^3 - c_a x^2 + (c_a + c_a^t)x - (c_a^2 + c_a^t - 2c_a + 2)$ (see Lemma 2.5.9) over $\mathbb{F}_q$ and find a root in $\mathbb{F}_{q^3}$; without loss of generality, suppose that this root is $\tilde{c}_a$. Next we construct the minimal polynomial of g^a over $\mathbb{F}_{q^3}$, i.e., $f_{g^a,3}(x) = x^2 - \tilde{c}_a x + 1$ and find a root of $f_{g^a,3}(x)$ in $\mathbb{F}_{q^6}$. Hence we obtain g^a or one of its conjugates over $\mathbb{F}_q$. The decompression can be achieved at a cost of $4A_1 + 1a_1 + 1F_1 + 1S_1 + 2m_1 + 1SR_{3,1} + 1SR_{2,3}$.

Now, to exponentiate $g^a \in \mu_n$ (or one of its conjugates over $\mathbb{F}_q$) to the power b, one first determines the *width-w radix-3 NAF* representation of b, i.e., $b = \sum_{i=0}^{\ell} b_i 3^i$ where $b_\ell > 0$, each nonzero b_i is nonzero modulo 3, and $|b_i| \leq (3^w - 1)/2$. The width-w radix-3 NAF representation of b contains on average $2\ell/(2w+1)$ nonzero digits (see [86] for more details on width-w radix-3 NAF representations). After precomputing and storing some elements one can compute g^{ab} at an average cost of l cubings and $2l/(2w+1)$ multiplications in $\mathbb{F}_{q^6}$. Using Karatsuba's technique, multiplying two elements in $\mathbb{F}_{q^6}$ can be accomplished with 18 multiplications in $\mathbb{F}_q$. Cubing in μ_n can be performed at a cost of $6C_1$. Note that by choosing a suitable polynomial for the extension $\mathbb{F}_{q^6}/\mathbb{F}_q$, we may ignore the costs of polynomial reductions in the extension field arithmetic.

Algorithm 2.4 A single-exponentiation algorithm based on LUC-exponentiation
Input: c_1 and a
Output: c_a

Write $a = \sum_{i=0}^{\ell-1} a_i 3^i$ where $a_i \in \{-1, 0, 1\}$ and $a_{\ell-1} = 1$
$C \leftarrow 1/(c_1^2 - 1)$
$\tilde{c}_1 \leftarrow$ a root of the polynomial $x^3 - c_1 x^2 + (c_1 + c_1^t)x - (c_1^2 + c_1^t + c_1 + 2)$
$\tilde{C} \leftarrow \tilde{c}_1^3 + \tilde{c}_1$
$j \leftarrow 1$
$s_u^{(j)} = [\tilde{c}_{u-1}, \tilde{c}_u] \leftarrow [2, \tilde{c}_1]$
for i from $\ell - 2$ down to 0 **do**
  $\tilde{c}_{3u} \leftarrow \tilde{c}_u^3$
  if $j = 0$ **then**
    $\{s_u^{(0)} = [\tilde{c}_u, \tilde{c}_{u+1}]$ is active$\}$
    if $a_i = -1$ **then**
      $\tilde{c}_{3u-1} \leftarrow C(\tilde{C}\tilde{c}_u^3 - \tilde{c}_{u+1}^3)$ {see (2.5.8)}
      $s_u^{(j)} \leftarrow [\tilde{c}_{3u-1}, \tilde{c}_{3u}]$
    else if $a_i = 0$ **then**
      $\tilde{c}_{3u+1} \leftarrow C(\tilde{c}_{u+1}^3 + \tilde{c}_1\tilde{c}_u^3)$ {see (2.5.10)}
      $s_u^{(j)} \leftarrow [\tilde{c}_{3u}, \tilde{c}_{3u+1}]$
    else if $a_i = 1$ **then**
      $\tilde{c}_{3u+1} \leftarrow C(\tilde{c}_{u+1}^3 + \tilde{c}_1\tilde{c}_u^3)$ {see (2.5.10)}
      $j \leftarrow 1$
      $s_u^{(j)} \leftarrow [\tilde{c}_{3u}, \tilde{c}_{3u+1}]$
    end if
  else
    $\{s_u^{(1)} = [\tilde{c}_{u-1}, \tilde{c}_u]$ is active$\}$
    if $a_i = -1$ **then**
      $\tilde{c}_{3u-1} \leftarrow C(\tilde{c}_{u-1}^3 + \tilde{c}_1\tilde{c}_u^3)$ {see (2.5.7)}
      $j \leftarrow 0$
      $s_u^{(j)} \leftarrow [\tilde{c}_{3u-1}, \tilde{c}_{3u}]$
    else if $a_i = 0$ **then**
      $\tilde{c}_{3u-1} \leftarrow C(\tilde{c}_{u-1}^3 + \tilde{c}_1\tilde{c}_u^3)$ {see (2.5.7)}
      $s_u^{(j)} \leftarrow [\tilde{c}_{3u-1}, \tilde{c}_{3u}]$
    else if $a_i = 1$ **then**
      $\tilde{c}_{3u+1} \leftarrow C(\tilde{C}\tilde{c}_u^3 - \tilde{c}_{u-1}^3)$ {(see 2.5.9)}
      $s_u^{(j)} \leftarrow [\tilde{c}_{3u}, \tilde{c}_{3u+1}]$
    end if
  end if
end for
Return $(\tilde{c}_u + \tilde{c}_u^q + \tilde{c}_u^{q^2})$

Hence, the average cost of computing c_{ab} can be approximated as $1SR_{3,1} + 1SR_{2,3} + (6C_1 + \frac{36}{(2w+1)}M_1)\ell$.

Direct exponentiation in μ_n without decompressing (Algorithm BPV-I)

This algorithm is based on the idea of Brouwer, Pellikaan and Verheul (see [13] or Section 2.3.2). Suppose c_a and an integer $b = \sum_{i=0}^{\ell} b_i 3^i$, where $b_i \in \{0, 1, 2\}$, in base-3 representation is given. By Lemma 2.5.10, we can determine $\{\tilde{c}_a, \tilde{d}_a\}$, the set of traces of g^a and g^{aq} over $\mathbb{F}_{q^2}$. Then the minimal polynomials of g^a and g^{aq} over $\mathbb{F}_{q^2}$ are $f_{g^a,2}(x) = x^3 - \tilde{c}_a x^2 + \tilde{c}_a^q x - 1$ and $f_{g^{aq},2}(x) = x^3 - \tilde{d}_a x^2 + \tilde{d}_a^q x - 1$. Without loss of generality let's assume that we know $f_{g^a,2}(x)$ (which can be computed at a cost of $1F_1 + 1F_2 + 1m_1 + 1SR_{2,1}$). That is, we have a copy of $\mathbb{F}_{q^6} = \mathbb{F}_{q^2}[x]/(f_{g^a,2}(x))$ and next we compute x^b modulo $f_{g^a,2}(x)$ using the repeated cube-and-multiply algorithm. Since

$$(\tau_2 x^2 + \tau_1 x + \tau_0)^3 = \tau_2^3 x^6 + \tau_1^3 x^3 + \tau_0^3,$$

each cubing (modulo $f_{g^a,2}(x)$) can be achieved at a cost of $4A_2 + 3C_2 + 5M_2$. Now, since

$$(\tau_2 x^2 + \tau_1 x + \tau_0)x = (\tau_2 \tilde{c}_a + \tau_1)x^2 + (2\tau_2 \tilde{c}_a^q + \tau_0)x + \tau_2,$$

multiplying by x can be achieved at a cost of $2A_2 + 1m_2 + 2M_2$. Similarly, we can show that multiplying by x^2 can be achieved at a cost of $3A_2 + 2m_2 + 4M_2$. Therefore, computing $x^b = w_2 x^2 + w_1 x + w_0$ with $w_i \in \mathbb{F}_{q^2}$ costs on average $(4A_2 + 3C_2 + 5M_2 + \frac{1}{3}(5A_2 + 3m_2 + 6M_2))(\ell - 1)$. Now,

$$c_{ab} = \mathrm{Tr}(x^b) = \mathrm{Tr}(w_2 x^2 + w_1 x + w_0) = \mathrm{Tr}(w_2 \mathrm{Tr}_2(x^2) + w_1 \mathrm{Tr}_2(x)).$$

Also note that $\mathrm{Tr}_2(x) = \tilde{c}_a$, and $\mathrm{Tr}_2(x^2) = \tilde{c}_a^2 + \tilde{c}_a^q$ follows from $x^3 = \tilde{c}_a x^2 - \tilde{c}_a^q x + 1$ in $\mathbb{F}_{q^6} = \mathbb{F}_{q^2}[x]/(f_{g^a,2}(x))$. Hence, we can write

$$c_{ab} = w_1 \tilde{c}_a + (w_1 \tilde{c}_a)^q + w_2(\tilde{c}_a^2 + \tilde{c}_a^q) + (w_2(\tilde{c}_a^2 + \tilde{c}_a^q))^q$$

and the expected cost of computing c_{ab} is $1SR_{2,1} + (\frac{17}{3}A_2 + 3C_2 + 7M_2))(\ell - 1)$ (we ignore the cost $(1F_1 + 1F_2 + 1m_1)$ that is dominated by $SR_{2,1}$ in the precomputation steps, and the costs $(3m_2)(\ell - 1)$ in the main loop and $(2F_2 + 2M_2)$ in the last step).

Direct exponentiation in μ_n without decompressing (Algorithm BPV-II)

The idea of the algorithm is similar to Algorithm BPV-I except that we work with a minimal polynomial over $\mathbb{F}_{q^3}$ instead of $\mathbb{F}_{q^2}$. Suppose c_a and an integer $b = \sum_{i=0}^{\ell} b_i 3^i$ in width-w radix-3 representation is given. By Lemma 2.5.9, we can determine the set of minimal polynomials of g^{aq^i} for $i = 0, 1, 2$. Without loss of generality let's assume that we

know $f_{g^a,3}(x) = x^2 - \tilde{c}_a x + 1$ (which can be computed at a cost of $4A_1 + 1a_1 + 1F_1 + 1S_1 + 2m_1 + 1SR_{3,1}$). That is, we have a copy of $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[x]/(f_{g^a,3}(x))$ and next we compute x^b modulo $f_{g^a,3}(x)$ using width-w radix-3 NAF exponentiation. Since

$$(\tau_1 x + \tau_0)^3 \equiv \tau_1^3(\tilde{c}_a^2 - 1)x + (\tilde{c}_a\tau_1^3 + \tau_0^3) \pmod{f_{g^a,3}(x)},$$

the cubing step can be achieved at a cost of $1A_1+1a_1+2C_3+2M_3+1m_3$. Using Karatsuba's technique, we can show that multiplying two elements in $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[x]/(f_{g^a,3}(x))$ can be accomplished at a cost of $4M_3$. Therefore, computing $x^b = w_1 x + w_0$, $w_i \in \mathbb{F}_{q^3}$, costs $(1A_1 + 1a_1 + 2C_3 + 2M_3 + 1m_3 + \frac{8}{(2w+1)}M_3)\ell$. Hence, the expected cost of computing

$$\begin{aligned} c_{ab} &= \mathrm{Tr}(x^b) = \mathrm{Tr}(w_1 x + w_0) \\ &= \mathrm{Tr}(w_1 \mathrm{Tr}_3(x) + \mathrm{Tr}_3(w_0)) = \mathrm{Tr}(w_1\tilde{c}_a + 2w_0) \\ &= w_1\tilde{c}_a + (w_1\tilde{c}_a)^q + (w_1\tilde{c}_a)^{q^2} \end{aligned}$$

is $1SR_{3,1}+(1A_1+2C_3+2M_3+\frac{8}{(2w+1)}M_3)\ell$ (we ignore the cost $(4A_1+1a_1+1F_1+1S_1+2m_1)$ that is dominated by $SR_{3,1}$ in the precomputation steps, and the costs $(1a_1+1m_3)\ell$ in the main loop and $(2A_3 + 2F_3 + 1M_3)$ in the last step).

Direct exponentiation in μ_n without decompressing (Algorithm BPV-III)

The idea of the algorithm is similar to Algorithm BPV-I and BPV-II except that we work with a minimal polynomial over $\mathbb{F}_q$ instead of $\mathbb{F}_{q^2}$ or $\mathbb{F}_{q^3}$. Given c_a and $b = \sum_{i=0}^{\ell} b_i 3^i$ in width-w radix-3 representation, we first determine $f_{g^a}(x) = x^6 - c_a x^5 + (c_a^t + c_a)x^4 - (c_a^2 + c_a^t + 2)x^3 + (c_a^t + c_a)x^2 - c_a x + 1$ at a cost of $1S_1$ (we ignore the cost $2A_1 + 1a_1 + 1F_1 + 2m_1$). Now, we have a copy of $\mathbb{F}_{q^6} = \mathbb{F}_q[x]/(f_{g^a}(x))$, and next we compute x^b modulo $f_{g^a}(x)$ using width-w radix-3 NAF exponentiation. Since

$$(\tau_5 x^5 + \tau_4 x^4 + \tau_3 x^3 + \tau_2 x^2 + \tau_1 x + \tau_0)^3 = \tau_5^3 x^{15} + \tau_4^3 x^{12} + \tau_3^3 x^9 + \tau_2^3 x^6 + \tau_1^3 x^3 + \tau_0^3,$$

cubing modulo $f_{g^a}(x)$ costs at least $6C_1 + 21M_1$. Moreover, using Karatsuba's technique, multiplying two elements in $\mathbb{F}_{q^6}$ appears to require at least $18M_1$ in $\mathbb{F}_q$. Hence computing $c_{ab} = \mathrm{Tr}(x^b)$ costs *at least* $2A_1 + 1F_1 + 1S_1 + (6C_1 + 21M_1 + \frac{36}{(2w+1)}M_1)\ell$.

2.6 Comparisons

2.6.1 Factor-4 compression

We compare the running times of the five exponentiation algorithms presented in Section 2.4. We first analyze the costs $SR_{2,1}$ and $SR_{2,2}$. Recall that $SR_{2,1}$ is the cost of

finding a root of the irreducible polynomial $\tilde{f}(x) = x^2 + cx + c^t$ for some $c \in \mathbb{F}_q$, and $SR_{2,2}$ is the cost of finding a root of the irreducible polynomial $\tilde{f}(x) = x^2+cx+1$ for some $c \in \mathbb{F}_{q^2}$. Shoup [79] showed that if a square-free degree-m polynomial over $\mathbb{F}_{p^l}$, where p is a small prime, is known to factor into d distinct irreducible same-degree polynomials over $\mathbb{F}_{p^l}$, then the factorization of that polynomial can be obtained at a cost of $\tilde{O}(m(dl)^2)$ $\mathbb{F}_{p^l}$-operations. Therefore, we may set $SR_{2,1} = \tilde{O}(32(\log_2 q)^2)$ $\mathbb{F}_{q^2}$-operations, and $SR_{2,2} = \tilde{O}(128(\log_2 q)^2)$ $\mathbb{F}_{q^4}$-operations.

Table 2.1: Comparison of exponentiation algorithms for factor-4 compression. The exponent is an ℓ-bit integer.

Algorithms	Precomputation	Main Loop
Algorithm 2.1	$1I_1 + 1M_1$	$(4M_1 + 4S_1)(\ell - 1)$
Algorithm 2.2	$1SR_{2,1}$	$(1M_2 + 1S_2)(\ell - 1)$
DDE (Section 2.4.3)	$1SR_{2,1} + 1SR_{2,2}$	$(\frac{9}{(w+1)}M_1 + 4S_1)\ell$
BPV-I (Section 2.4.3)	$1SR_{2,1}$	$(\frac{3}{2}M_2 + 2S_2)(\ell - 1)$
BPV-II (Section 2.4.3)	$1F_1$	$(7M_1 + 4S_1)(\ell - 1)$

We can conclude from Table 2.1 that, for a suitable choice of w, Algorithm DDE is the fastest algorithm if the precomputations are done in advance (which is the case if the base is fixed). Otherwise, Algorithm 2.1 is the fastest one.

Remark 2.6.1. It is possible to obtain better running time estimates (at least for the running times of the main loops) for some of the algorithms listed in Table 2.1. Adapting the double-exponentiation technique given in [84] to speed up Algorithm 2.2, one can estimate the running time of Algorithm 2.2 as $2SR_{2,1} + (0.75M_2)\ell$. Using simultaneous multi-exponentiation or window-NAF techniques one can optimize the running times of Algorithms BPV-I and BPV-II. These techniques do not seem to reduce the cost of the squaring steps in the main loops of Algorithm BPV-I and BPV-II (which are $(1M_2 + 2S_2)(\ell - 1)$ and $(6M_1 + 4S_1)(\ell - 1)$, respectively) other than transferring some of the cost to the precomputation phase. The mixed-multiplication method (see [39]) can be adapted to speed up Algorithm DDE given input values c_a and b. More precisely, after decompressing c_a to g^a, one can first compute $g^{ab}h$ for some $h \in \mathbb{F}^*_{q^2}$ at an approximate cost of $(\frac{2}{(w+1)}M_2)\ell$ instead of computing g^{ab} using Karatsuba's multiplication technique at an approximate cost of $(\frac{9}{(w+1)}M_1)\ell$ (see Section 2.4.3), Then c_{ab} can be computed by taking the square root of $\mathrm{Tr}((g^{ab}h)^{q^2-1}) = \mathrm{Tr}(g^{ab(q^2-1)}) = c_{2ab}$. Using this method, the cost of Algorithm DDE can be approximated as $1SR_{2,1} + 1SR_{2,2} + (\frac{6}{(w+1)}M_1)\ell + 1I_4$; the cost of this faster algorithm is given in Table 2.2. Hence, we still expect Algorithm 2.1 to be the

fastest algorithm for general bases and, if the base is fixed, Algorithm DDE to be faster than Algorithm 2.1 and Algorithm 2.2 for a suitable choice of w.

To be more concrete, we list the expected running times of the five exponentiation algorithms in a particular setting in Table 2.2 based on the estimates given in Table 2.1. For the 128-bit security level, we let $q = 2^{1223}$ and $t = 2^{612}$. Then $q+1+t = 5n$ where n is a 1221-bit prime. We will ignore the costs F_i, S_i and, using Karatsuba's technique, assume $M_2 = 3M_1$. We also select $w = 5$.

Table 2.2: Comparison of exponentiation algorithms for factor-4 compression at the 128-bit security level. The exponent is an 1221-bit integer.

Algorithms	Multiplication cost for general bases	Multiplication cost for fixed bases
Algorithm 2.1	$1I_1 + 4881M_1$	$4880M_1$
Algorithm 2.2	$1SR_{2,1} + 3660M_1$	$3660M_1$
DDE (Section 2.4.3, Remark 2.6.1)	$1SR_{2,1} + 1SR_{2,2} + 1I_4 + 1221M_1$	$1I_4 + 1221M_1$
BPV-I (Section 2.4.3)	$1SR_{2,1} + 5490M_1$	$5490M_1$
BPV-II (Section 2.4.3)	$8540M_1$	$8540M_1$

2.6.2 Factor-6 compression

We compare the running times of the six algorithms presented in Section 2.5 and the XTR_3 [78] algorithm. We note that even though the running time of XTR_3 is estimated as $1SR_{2,1} + (8M_1)\ell$ in [78], adapting the double-exponentiation technique for speeding up XTR (see [84]) to XTR_3 one can roughly approximate the cost of XTR_3 as $2SR_{2,1}+(3M_1)\ell$ which we use in Table 2.3.

We first analyze the costs $SR_{2,1}$, $SR_{2,3}$ and $SR_{3,1}$. Similarly as in Section 2.6.1, using Shoup's method [79] we may let $SR_{3,1} = \tilde{O}(27(\log_2 q)^2)$ $\mathbb{F}_{q^3}$-operations. Now, we will consider $SR_{2,1}$ and $SR_{2,3}$, and see that these costs are negligible compared to $SR_{3,1}$. We let $q = 3^{2r+1}$ and recall that $SR_{2,1}$ is the cost of finding a root of the irreducible polynomial $\tilde{f}(x) = x^2 - cx + c^t$ over $\mathbb{F}_q$. The roots of this polynomial are $2(c \pm \sqrt{c^2 + 2c^t})$. Since $\tilde{f}(x)$ is irreducible over $\mathbb{F}_q$, $C = c^2 + 2c^t$ must be a quadratic non-residue in $\mathbb{F}_q$. Moreover, since $q \equiv 3 \pmod 4$, -1 is a quadratic non-residue in $\mathbb{F}_q$. Therefore, $-C$ is a quadratic residue in $\mathbb{F}_q$ and finding a root of $\tilde{f}(x)$ reduces to finding a square root of $-C$ in $\mathbb{F}_q$. Namely, the roots of $\tilde{f}(x)$ are $2(c \pm \sqrt{-C}\sqrt{-1})$ and also note that $\sqrt{-C} = (-C)^{\frac{q+1}{4}}$ and $\sqrt{-1} \in \mathbb{F}_{q^2}$.

Barreto et al. [6] observed that

$$\frac{q+1}{4} = 6\sum_{i=0}^{r-1}(3^2)^i + 1$$

and that computing

$$(-C)^{\frac{q+1}{4}} = \left((C^2)^{\sum_{i=0}^{r-1}(3^2)^i}\right)^3 (-C)$$

can be achieved at a cost of $1S_1 + (\lfloor \log_2 r \rfloor + HW(r))M_1$. Therefore, after computing $\sqrt{-1} \in \mathbb{F}_{q^2}$ we can compute a root of $\tilde{f}(x)$ in $\mathbb{F}_{q^2}$ at a cost of $1m_2 + 1M_2 + 1S_1 + (\lfloor \log_2 r \rfloor + HW(r)+1)M_1$. Hence, we may set $1SR_{2,1} = 1m_2 + 1M_2 + 1S_1 + (\lfloor \log_2 r \rfloor + HW(r)+1)M_1$. Similarly, we can show $1SR_{2,3} = 1m_6 + 1M_6 + 1S_3 + (\lfloor \log_2 (3r+1) \rfloor + HW(3r+1)+1)M_3$.

Table 2.3: Comparison of exponentiation algorithms for factor-6 compression. The exponent is an ℓ-bit integer.

Algorithms	Precomputation	Main Loop
Algorithm 2.3	$1I_1 + 2M_1 + 2S_1$	$(53A_1 + 6F_1 + 23M_1 + 6S_1)(\ell - 1)$
Algorithm 2.4	$1SR_{3,1}$	$(1A_3 + 2C_3 + 2M_3)(\log_3 2)(\ell - 1)$
DDE (Section 2.5.3)	$1SR_{3,1} + 1SR_{2,3}$	$(6C_1 + \frac{36}{(2w+1)}M_1)(\log_3 2)\ell$
BPV-I (Section 2.5.3)	$1SR_{2,1}$	$(\frac{17}{3}A_2 + 3C_2 + 7M_2)(\log_3 2)(\ell - 1)$
BPV-II (Section 2.5.3)	$1SR_{3,1}$	$(1A_1 + 2C_3 + 2M_3 + \frac{8}{(2w+1)}M_3)(\log_3 2)\ell$
BPV-III (Section 2.5.3)	$1S_1$	$\geq (6C_1 + 21M_1 + \frac{36}{2w+1}M_1)(\log_3 2)\ell$
XTR_3 ([78])	$2SR_{2,1}$	$(3M_1)\ell$

We can conclude from Table 2.3 that, for a suitable choice of w, Algorithm DDE is the fastest algorithm if the precomputations are done in advance (which is the case if the base is fixed). Otherwise, XTR_3 is the fastest one.

Remark 2.6.2. It is possible to obtain better running time estimates (at least for the running times of the main loops) for some of the algorithms listed in Table 2.3. Adapting the double-exponentiation technique given in [84] to speed up Algorithm 2.4, one can estimate the running time of Algorithm 2.4 as $2SR_{3,1} + (0.75M_3 + 0.17S_3)\ell$. Using simultaneous multi-exponentiation or window-NAF techniques one might optimize the running times of Algorithms BPV-I, BPV-II and BPV-III. These techniques do not seem to reduce the cost of the cubing steps in the main loops of Algorithms BPV-I, BPV-II and BPV-III (which are roughly $(5M_2)(\log_3 2)\ell \approx (6.31M_1)\ell$, $(2M_3)(\log_3 2)\ell \approx (7.57M_1)\ell$ and $(21M_1)(\log_3 2)\ell \approx (13.25M_1)\ell$, respectively) other than transferring some of the cost to the precomputation phase. The mixed-multiplication method (see [39]) can be adapted to

speed up Algorithm DDE by slightly changing its output value from c_{ab} to c_{2ab} given input values c_a and b (we note that Algorithm DDE with this slight change in its output can still be used in the cryptographic applications mentioned in Section 2.1). More precisely, after decompressing c_a to g^a, one can first compute $g^{ab}h$ for some $h \in \mathbb{F}_{q^3}^*$ at an approximate cost of $(\frac{4}{(2w+1)}M_3)(\log_3 2)\ell$ instead of computing g^{ab} using Karatsuba's multiplication technique at an approximate cost of $(\frac{36}{(2w+1)}M_1)(\log_3 2)\ell$ (see Section 2.5.3). Then $\mathrm{Tr}((g^{ab}h)^{q^3-1}) = \mathrm{Tr}(g^{ab(q^3-1)}) = c_{2ab}$ can be computed at an approximate cost of $1I_6$. Using this method, the cost of Algorithm DDE can be approximated as $1SR_{3,1}+1SR_{2,3}+(\frac{24}{(2w+1)}M_1)(\log_3 2)\ell+1I_6$; the cost of this faster algorithm is given in Table 2.4. Hence, we still expect XTR_3 to be the fastest algorithm for general bases and, if the base is fixed, Algorithm DDE to be significantly faster than the other algorithms for a suitable choice of w.

To be more concrete, we list the expected running times of the seven exponentiation algorithms in a particular setting in Table 2.4 based on the estimates given in Table 2.3. For the 128-bit security level, we let $q = 3^{509}$ and $t = 3^{255}$. Then $q+1-t = 7n$ where n is an 804-bit prime. We will ignore the costs A_i, a_i, C_i, F_i, m_i and, using Karatsuba's technique, assume $M_2 = 3M_1$, $M_3 = 6M_1$ and $M_6 = 18M_1$. We also assume $S_i = M_i$, and select $w = 5$. Note that $r = 254$, $\lfloor \log_2 r \rfloor = 8$, $\lfloor \log_2 (3r+1) \rfloor = 9$, $HW(r) = 7$, $HW(3r+1) = 8$, and $1SR_{2,1} = 20M_1$ and $1SR_{2,3} = 132M_1$.

Table 2.4: Comparison of exponentiation algorithms for factor-6 compression at the 128-bit security level. The exponent is an 804-bit integer.

Algorithms	Multiplication cost for general bases	Multiplication cost for fixed bases
Algorithm 2.3	$1I_1 + 23291M_1$	$23287M_1$
Algorithm 2.4	$1SR_{3,1} + 6080M_1$	$6080M_1$
DDE (Section 2.5.3, Remark 2.6.2)	$1SR_{3,1} + 1I_6 + 1239M_1$	$1I_6 + 1107M_1$
BPV-I (Section 2.5.3)	$10660M_1$	$10640M_1$
BPV-II (Section 2.5.3)	$1SR_{3,1} + 8301M_1$	$8301M_1$
BPV-III (Section 2.5.3)	$\geq 12314M_!$	$\geq 12313M_1$
XTR_3 ([78])	$2452M_1$	$2412M_1$

2.7 Non-uniqueness of factor-6 compression in large characteristic

Barreto and Naehrig [7, Section 3] suggested that pairing values for the asymmetric pairing derived from Barreto-Naehrig (BN) elliptic curves can be compressed to one-sixth of their length. BN pairing values lie in the subgroup $\mu_n \subset \mathbb{F}_{p^{12}}^*$ where $n = n(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1$, $p = p(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$, and $x \in \mathbb{Z}$ is such that $n(x)$ and $p(x)$ are prime. Their compression method identifies the elements of the subgroup μ_n of $\mathbb{F}_{q^6}^*$ (where $q = p^2$) with their traces over $\mathbb{F}_q$. In fact, given $q = p^2$ and n as above, one can write $t = q + 1 - h \cdot n$ where t is the trace of the corresponding BN curve over $\mathbb{F}_q$, $p \nmid t$, and show that μ_n has embedding degree 6 over $\mathbb{F}_q$ with trace t. However, as one can see from Lemma 2.5.1, $\mathrm{Tr}(g)$ does not suffice to identify an element $g \in \mu_n$ uniquely up to its conjugates over $\mathbb{F}_q$. Hence, it is possible that there are *collisions* for the trace function. That is, there may exist elements $h_1, h_2 \in \mu_n \subset \mathbb{F}_{q^6}^*$ such that h_1 and h_2 are not conjugates of each other over $\mathbb{F}_q$ and $\mathrm{Tr}(h_1) = \mathrm{Tr}(h_2)$, in which case the Barreto-Naehrig compression method fails.

We searched for collisions in the case $x \in \{-41, -15, -7, -3, -2, -1, 1, 5, 6, 7, 20\}$ and discovered one when $x = 6$ (where $n = 55117$ and $q = (55333)^2$). We list below one BN and eight BN-like sets of parameters (p, n, T, g, u, v) such that $n \mid (p^4 - p^2 + 1)$, $\mu_n = \langle g \rangle$ is the order-n subgroup of $\mathbb{F}_{q^6}^*$, and (g^u, g^v) is a collision with colliding value $T = \mathrm{Tr}(g^u) = \mathrm{Tr}(g^v)$. The first example corresponds to the parameter of a BN curve. The subgroups μ_n in examples (2)–(5) can be realized as the images of pairing functions defined on elliptic curves over $\mathbb{F}_p$ having embedding degree 12; these elliptic curves are not BN curves. On the other hand, the subgroups μ_n in examples (6)–(9) cannot be realized as the image of pairing functions defined on elliptic curves over $\mathbb{F}_p$ since n is not in the Hasse interval $[(\sqrt{p} - 1)^2, (\sqrt{p} + 1)^2]$.

1. $(55333, 55117, 45541, g, 2583, 6758)$ where
   $\mathbb{F}_q = \mathbb{F}_p[z]/(z^2 + 2)$,
   $\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + 51894z + 9346)$,
   $g = (5638z + 51877)w^5 + (13297z + 52777)w^4 + (20924z + 25318)w^3 + (12991z + 51370)w^2 + (12014z + 15762)w + 15570z + 33355$.

2. $(113, 97, 46, g, 20, 29)$ where
   $\mathbb{F}_q = \mathbb{F}_p[z]/(z^2 + 101z + 3)$,
   $\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + 112z)$,
   $g = (77z+47)w^5+(29z+36)w^4+(52z+24)w^3+(58z+14)w^2+(70z+19)w+35z+49$.

3. $(297457, 296941, 170970, g, 42243, 120695)$ where
$\mathbb{F}_q = \mathbb{F}_p[z]/(z^2+5)$,
$\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + 226781z + 185746)$,
$g = (16282z+114368)w^5+(264807z+131493)w^4+(52866z+175278)w^3+(153254z+81017)w^2 + (55521z + 87692)w + 27500z + 23791$.

4. $(757363, 758053, 147442, g, 195883, 532217)$ where
$\mathbb{F}_q = \mathbb{F}_p[z]/(z^2+1)$,
$\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + 538552z + 38070)$,
$g = (155890z+54538)w^5+(593065z+407753)w^4+(421831z+252766)w^3+(260748z+405992)w^2 + (426293z + 142508)w + 240615z + 248519$.

5. $(758743, 758053, 181973, g, 26808, 304248)$ where
$\mathbb{F}_q = \mathbb{F}_p[z]/(z^2+1)$,
$\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + 205464z + 531602)$,
$g = (644749z+587471)w^5+(152111z+593113)w^4+(218499z+561168)w^3+(605298z+638869)w^2 + (634695z + 684366)w + 667616z + 318403$.

6. $(107873, 100333, 10836z + 78750, g, 6775, 11682)$ where
$\mathbb{F}_q = \mathbb{F}_p[z]/(z^2+3)$,
$\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + 70681z + 104404)$,
$g = (74900z + 35768)w^5 + (67288z + 57726)w^4 + (107242z + 94650)w^3 + (31629z + 3630)w^2 + (87341z + 35135)w + 64176z + 45731$.

7. $(107873, 100333, 97037z + 78750, g, 14995, 20801)$ where
$\mathbb{F}_q = \mathbb{F}_p[z]/(z^2+3)$,
$\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + 70681z + 104404)$,
$g = (71680z + 68567)w^5 + (99591z + 66980)w^4 + (34944z + 30340)w^3 + (8164z + 61554)w^2 + (29313z + 44640)w + 33137z + 62160$.

8. $(147347, 135193, 56095z + 80249, g, 4989, 12193)$ where
$\mathbb{F}_q = \mathbb{F}_p[z]/(z^2+1)$,
$\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + 76671z + 35636)$,
$g = (120469z + 82203)w^5 + (77634z + 4734)w^4 + (127289z + 74128)w^3 + (106306z + 13444)w^2 + (82983z + 115891)w + 34710z + 136734$.

9. $(147347, 135193, 91252z + 80249, g, 3676, 12104)$ where
   $\mathbb{F}_q = \mathbb{F}_p[z]/(z^2 + 1)$,
   $\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + 76671z + 35636)$,
   $g = (113834z + 48691)w^5 + (87284z + 70855)w^4 + (85568z + 73528)w^3 + (102712z + 53673)w^2 + (13537z + 46246)w + 105305z + 14472$.

2.8 Concluding remarks

We have shown how to compress, by a factor of 4, pairing values of the commonly-used symmetric bilinear pairings over characteristic-two fields, and also further explored compressing pairing values of symmetric bilinear pairings over characteristic-three fields by a factor of 6. We have shown how one can exponentiate using the compressed pairing values. Our exponentiation algorithms are reasonably efficient. In particular, if the base is fixed then we expect at least a 54% speed up over the fastest previously known algorithm XTR_3 for the factor-6 compression case.

Chapter 3

Double Exponentiation in Compressed Cyclotomic Subgroups

In Chapter 2, using the trace representation of elements, we showed how to compress elements of subgroups $G \subset \mathbb{F}_{q^4}^*$ of orders $q \pm \sqrt{2q} + 1$, where $q = 2^m$, m is odd, by a factor of 4. We also showed that single-exponentiation can be efficiently performed using compressed representations. In this chapter we show that double-exponentiation can be efficiently performed using factor-4 compressed representation of elements. In addition to giving a considerable speed up to the fastest single-exponentiation algorithm for general bases (see Algorithm 2.1 in Chapter 2), double-exponentiation can be used to adapt our compression technique to ElGamal-type signature schemes. The results of this chapter appeared in [49].

The remainder of this chapter is organized as follows. In Section 3.1 we review our work on factor-4 compression presented in Chapter 2. Section 3.2 presents our double-exponentiation algorithm and an analysis of the algorithm. Two cryptographic applications of our double-exponentiation algorithm are given in Section 3.3. We make some concluding remarks in Section 3.4. Throughout this chapter we use the same terminology and notation introduced in Section 2.2.

3.1 A review of factor-4 compression

Let r be a positive integer, and let $q = 2^{2r+1}$, $t = \pm 2^{r+1}$, $T = |t|$. The values of r for which $q + 1 - t = hn$ and n is prime lead to a multiplicative subgroup μ_n of $\mathbb{F}_{q^4}^*$ of prime order n with embedding degree 4 and trace t. We fix h, n, q, t, T and $\mu_n = \langle g \rangle$ in this fashion, and also write $c_u = \mathrm{Tr}(g^u)$. Note that $c_0 = 0$ and $c_{uT} = c_u^T$.

Table 3.1: Cost of Algorithm 2.1 (page 21) for factor-4 compression. The exponent is an ℓ-bit integer.

Algorithm	Precomputation	Main Loop
Algorithm 2.1	$1I_1 + 1M_1$	$(4M_1 + 4S_1)(\ell - 1)$

The following theorem shows that $g^u \in \mu_n$ can be uniquely represented (up to conjugation over $\mathbb{F}_q$) by its trace, thus providing compression by a factor 4. From now on, we will refer to this group $\mu_n \subset \mathbb{F}_{q^4}^*$ as the *factor-4 subgroup* of $\mathbb{F}_{q^4}^*$ with trace t.

Theorem 3.1.1 (restatement of Corollary 2.4.5)**.** *Let $\mu_n = \langle g \rangle$ be the factor-4 subgroup of $\mathbb{F}_{q^4}^*$ with trace t. Let $f_{g^u}(x)$ be the minimal polynomial of $g^u \in \mu_n$ over $\mathbb{F}_q$. Then*

$$f_{g^u}(x) = x^4 + c_u x^3 + c_u^T x^2 + c_u x + 1.$$

Next, we recall some facts from Chapter 2 that we will use in Section 3.2.

Lemma 3.1.2. *Let $\mu_n = \langle g \rangle$ be the factor-4 subgroup of $\mathbb{F}_{q^4}^*$ with trace t. Then for all integers u and v we have*
(i) $c_{u+v} = (c_u + c_{u-2v})c_v + c_{u-v}c_v^T + c_{u-3v}$ (restatement of Corollary 2.4.4(i)).
(ii) $c_u = c_{-u}$ (restatement of Lemma 2.4.2(i)).
(iii) $c_{2u} = c_u^2$ (restatement of Corollary 2.4.4(ii)).
(iv) $c_u c_v = c_{u+v} + c_{u-v} + c_{u+v(t-1)} + c_{v+u(t-1)}$ (restatement of Lemma 2.4.2(ii)).

Remark 3.1.3. Throughout the remainder of this chapter, we will assume without loss of generality that the trace t is positive. If t is negative then one can replace the expressions of the form $c_u^{p(t)}$, where p is some polynomial, by $c_u^{p(T)}$ without changing the validity of the results in this chapter.

Let a, b be integers with $0 < a, b < n$. Single-exponentiation to the base a in μ_n is the operation of computing c_{ab} given c_a and b. Five single-exponentiation algorithms were presented in Chapter 2. We concentrate on Algorithm 2.1 (page 21) which works directly with c_a, and is the fastest exponentiation algorithm for general bases.

Next, we give a generalization of Theorem 4.7 in Chapter 2 that will be used in Section 3.2 to describe a double-exponentiation algorithm in μ_n.

Theorem 3.1.4. *Let $\mu_n = \langle g \rangle$ be the factor-4 subgroup of $\mathbb{F}_{q^4}^*$ with trace t. Let $c_u = \mathrm{Tr}(g^u)$,*

$$A = \begin{pmatrix} c_v & c_v & 0 & 0 \\ 0 & c_v & c_v & 0 \\ 0 & 1 & c_v^t & 1 \\ 1 & c_v^t & 1 & 0 \end{pmatrix}, \; X = \begin{pmatrix} c_{2u-3v} \\ c_{2u-v} \\ c_{2u+v} \\ c_{2u+3v} \end{pmatrix}, \; Y = \begin{pmatrix} (c_u + c_{u-2v})^2 + c_{u-v}^2 c_v^t \\ (c_{u+v} + c_{u-v})^2 + c_u^2 c_v^t \\ (c_u + c_{u+v})^2 c_v \\ (c_{u-v} + c_u)^2 c_v \end{pmatrix}.$$

Then
(i) A is invertible and $AX = Y$.
(ii) If c_v is given then A and A^{-1} can be efficiently computed.
(iii) $c_{2u-v} = \frac{1}{c_v^{t+1}}\left((c_{u+v} + c_u + c_{u-v} + c_{u-2v})^2 + (c_u + c_{u-v})^2(c_v^t + c_v^2)\right)$.

Proof. The proof is analogous to the proof of Theorem 4.7 in Chapter 2. □

Corollary 3.1.5. *Let $\mu_n = \langle g \rangle$ be the factor-4 subgroup of $\mathbb{F}_{q^4}^*$ with trace t. Let c_v, c_{2u-v} and $s_{u,v} = [c_{u-2v}, c_{u-v}, c_u, c_{u+v}]$ be given. Then*
(i) $c_{u+2v} = (c_{u+v} + c_{u-v})c_v + c_u c_v^t + c_{u-2v}$, *and can be computed at a cost of* $1F_1 + 2M_1$.
(ii) $c_{2u+v} = (c_{u+v} + c_{u-v})c_u + c_v c_u^t + c_{2u-v}$, *and can be computed at a cost of* $1F_1 + 2M_1$.
(iii) $c_{u-3v} = (c_u + c_{u-2v})c_v + c_{u-v}c_v^t + c_{u+v}$, *and can be computed at a cost of* $1F_1 + 2M_1$.
(iv) $c_{3u-v} = (c_v + c_{2u-v})c_u + c_{u-v}c_u^t + c_{u+v}$, *and can be computed at a cost of* $1F_1 + 2M_1$.
(v) $c_{u-4v} = c_{u+2v} + (c_u + c_{u-2v})(c_v^2 + c_v^t + 1) + c_{u-v}c_v^{t+1}$, *and* (c_{u+2v}, c_{u-4v}) *can be computed at a cost of* $1F_1 + 5M_1 + 1S_1$.
(vi) $c_{4u-v} = c_{2u+v} + (c_v + c_{2u-v})(c_u^2 + c_u^t + 1) + c_{u-v}c_u^{t+1}$, *and* (c_{2u+v}, c_{4u-v}) *can be computed at a cost of* $1F_1 + 5M_1 + 1S_1$.
(vii) $c_{3u+v} = c_{3u-v} + (c_{u+v} + c_{u-v})(c_u^2 + c_u^t + 1) + c_v c_u^{t+1}$, *and* (c_{3u-v}, c_{3u+v}) *can be computed at a cost of* $1F_1 + 5M_1 + 1S_1$.

Proof. (i)–(iv) follow from Lemma 3.1.2(i) and (ii).
(v) Using Lemma 3.1.2(i) and (ii), we first write

$$c_{u+2v} = (c_{u+v} + c_{u-v})c_v + c_u c_v^t + c_{u-2v} \quad (3.1.1)$$
$$c_{u-4v} = (c_{u-v} + c_{u-3v})c_v + c_{u-2v}c_v^t + c_u. \quad (3.1.2)$$

Now, adding (3.1.1) and (3.1.2) together and replacing $c_{u-3v} = (c_u + c_{u-2v})c_v + c_{u-v}c_v^t + c_{u+v}$ gives $c_{u-4v} = c_{u+2v} + (c_u + c_{u-2v})(c_v^2 + c_v^t + 1) + c_{u-v}c_v^{t+1}$. Finally, once c_{u+2v} is computed as in (i) at a cost of $1F_1 + 2M_1$, it is clear that c_{u-4v} can be computed at a cost of $3M_1 + 1S_1$, which completes the proof.
(vi) The proof follows as in (v) after interchanging u and v.
(vii) The proof follows as in (v) after replacing (u, v) by $(u - v, u)$. □

3.2 Double-exponentiation in factor-4 groups

Definition 3.2.1. Let $\mu_n = \langle g \rangle$ be the factor-4 subgroup of $\mathbb{F}_{q^4}^*$ with trace t, and as usual let $c_u = \mathrm{Tr}(g^u)$. Let a, b, k, l be integers with $0 < a, b, k, l < n$. *Double-exponentiation* to the base (k, l) in μ_n is the operation of computing c_{ak+bl} given a, b, c_l and $s_{k,l} = [c_{k-2l}, c_{k-l}, c_k, c_{k+l}]$.

We assume that a, b, k, l are strictly positive as otherwise double-exponentiation just becomes a single-exponentiation. Note that k and l are not necessarily known.

A double-exponentiation algorithm was presented by Stam and Lenstra [84] for second-degree and third-degree recursive sequences. Their algorithm is an adaptation of Montgomery's method [68] to compute second-degree recursive sequences. In this section, we adapt the techniques used in [84, 68] and present a double-exponentiation algorithm for fourth-degree recursive sequences.

Algorithm 3.1 starts with $u = k, v = l, d = a > 0$, $e = b > 0$, from which it follows that $ud + ve = ak + bl = \tau$, c_v and $s_{u,v}$ are known, and c_{2u-v} can be computed at a cost of $1F_1 + 1I_1 + 3M_1 + 3S_1$ by Theorem 3.1.4. The reason we introduce the term c_{2u-v} in step 2 of Algorithm 3.1 is to avoid the repeated computation of c_{2u-v} during the updates. In the main part of the algorithm, u, v, d, e will be updated so that $ud + ve = ak + bl = \tau$ holds, $d, e > 0$, and $(d + e)$ decreases until $d = e$. Also, c_v, c_{2u-v} and $s_{u,v}$ are updated according to the new values of u and v. When $d = e$, we will have $\tau = d(u + v)$ and $s_{u,v} = [c_{u-2v}, c_{u-v}, c_u, c_{u+v}]$. Finally, we compute $c_\tau = c_{d(u+v)}$ using a single-exponentiation algorithm. Table 3.2 gives the update rules for $u, v, d, e, c_v, c_{2u-v}$ and $s_{u,v}$. Table 3.3 gives the cost of each update operation and the factor by which each update reduces $(d + e)$. The cost analysis as summarized in the third column of Table 3.3 follows from Table 3.2 and Corollary 3.1.5.

Table 3.2: Update rules for double-exponentiation in factor-4 groups

Rule	Condition	d	e	u	v	c_v	c_{2u-v}	$s_{u,v} = [c_{u-2v}, c_{u-v}, c_u, c_{u+v}]$
if $d \geq e$								
R1	$d \leq 4e$	$d - e$	e	u	$u + v$	c_{u+v}	c_{u-v}	$[c_{u+2v}, c_v, c_u, c_{2u+v}]$
R2	$d \equiv e \pmod 2$	$(d - e)/2$	e	$2u$	$u + v$	c_{u+v}	c_{3u-v}	$[c_v^2, c_{u-v}, c_u^2, c_{3u+v}]$
R3	$d \equiv 0 \pmod 2$	$d/2$	e	$2u$	v	c_v	c_{4u-v}	$[c_{u-v}^2, c_{2u-v}, c_u^2, c_{2u+v}]$
R4	$e \equiv 0 \pmod 2$	d	$e/2$	u	$2v$	c_v^2	c_{u-v}^2	$[c_{u-4v}, c_{u-2v}, c_u, c_{u+2v}]$
else								
S	$e > d$	e	d	v	u	c_u	c_{u-2v}	$[c_{2u-v}, c_{u-v}, c_v, c_{u+v}]$

Correctness: Let $m = \gcd(a, b)$, $a = ma'$, and $b = mb'$. It is easy to see that if $m = 2^r m'$, $r \geq 0$, and m' is odd then after step 6 in Algorithm 3.1 we will have $f = 2^r$, $d = m'a'$, $e = m'b'$. Moreover, after step 9, we will have $d = e = m'$ and $d(u + v) = m'a'k + m'b'l$ since $ud + ve$ is kept invariant while applying the rules in Table 3.2. Hence, $ak + bl = 2^r m'a'k + 2^r m'b'l = f \cdot d(u + v)$, $c_{ak+bl} = c_{f \cdot d(u+v)}$, and the correctness of the algorithm follows from Lemma 3.1.2(iii), as $c_{f \cdot d(u+v)} = c_{d(u+v)}^f$.

Table 3.3: Analysis of update rules for double-exponentiation in factor-4 groups

Rule	Condition	Cost	Reduction factor for $(d+e)$
if $d \geq e$			
R1	$d \leq 4e$	$2F_1 + 4M_1$	$\geq 5/4,\ < 2$
R2	$d \equiv e \pmod 2$	$1F_1 + 5M_1 + 2S_1$	2
R3	$d \equiv 0 \pmod 2$	$1F_1 + 5M_1 + 2S_1$	$\geq 5/3,\ < 2$
R4	$e \equiv 0 \pmod 2$	$1F_1 + 5M_1 + 2S_1$	$> 1,\ < 10/9$
else			
S	$e > d$	0	1

Algorithm 3.1 A double-exponentiation algorithm
Input: $a > 0, b > 0, c_l$ and $s_{k,l} = [c_{k-2l}, c_{k-l}, c_k, c_{k+l}]$
Output: c_{ak+bl}

1: $u \leftarrow k,\ v \leftarrow l,\ d \leftarrow a,\ e \leftarrow b,\ s_{u,v} \leftarrow s_{k,l}$
2: $c_{2u-v} \leftarrow \frac{1}{c_v^{t+1}} \left((c_{u+v} + c_u + c_{u-v} + c_{u-2v})^2 + (c_u + c_{u-v})^2 (c_v^t + c_v^2)\right)$
3: $f \leftarrow 1$
4: **while** d and e are both even **do**
5: $\quad d \leftarrow d/2,\ e \leftarrow e/2,\ f \leftarrow 2f$
6: **end while**
7: **while** $d \neq e$ **do**
8: $\quad$ Execute the first applicable rule in Table 3.2
9: **end while**
10: Compute $c_{d(u+v)}$ using Algorithm 2.1 with input c_{u+v} and d
11: Return $c_{d(u+v)}^f$

Analysis: The running time analysis of Algorithm 3.1 is similar to that in [68]. We will ignore the costs F_1 and S_1. If R4 is never required during the execution of Algorithm 3.1, then it is clear from Table 3.3 that the cost of the second while loop in Algorithm 3.1 never exceeds

$$\max(4\log_{5/4}(a'+b'), 5\log_2(a'+b'), 5\log_{5/3}(a'+b'))M_1 \approx 12.4\log_2(a'+b')M_1.$$

Now, suppose that R4 is used $i > 0$ times successively, with the starting (d, e) value (d_1, e_1). That is, $d_1 > 4e_1$, $d_1 \equiv 1 \pmod 2$, and $e_1 \equiv 0 \pmod 2$. Let (d_2, e_2) be the updated value of (d, e) after i applications of R4. Clearly, $d_1 = d_2$ and $e_1 = e_2 2^i$. Now, only the rule R2 is applicable, and suppose we apply R2 and R3 (possibly after R2) j times until R1 qualifies (i.e. $d \leq 4e$) or $j \leq i$; and suppose that the (d, e) value is updated to (d_3, e_3). Clearly,

$e_2 = e_3$ and $d_3 \leq d_2/2^i$. If $d_3 \leq 4e_3$ then

$$\frac{(d_1 + e_1)}{(d_3 + e_3)} \geq \frac{5e_1}{5e_3} = 2^i.$$

If $j = i$ then

$$\frac{(d_1 + e_1)}{(d_3 + e_3)} \geq \frac{d_2 + e_2 2^i}{d_2/2^j + e_2} = 2^i.$$

In both cases, the value $(d + e)$ value is reduced at least by a factor of 2^i at a cost of $5(i + j)M_1 \leq 10iM_1$. Hence, the total cost of the second while loop in Algorithm 3.1 never exceeds $12.4 \log_2 (a' + b')M_1$. Using Algorithm 2.1 for step 10 in Algorithm 3.1, we can conclude that the total cost of Algorithm 3.1 never exceeds $12.4 \log_2 (a + b)M_1$.

In our experiments we observed that the performance of Algorithm 3.1 in practice is remarkably better than the upper bound $12.4 \log_2 (a + b)M_1$. Moreover, the behavior of Algorithm 3.1 becomes very stable as $(a + b)$ gets larger. We tested the performance of Algorithm 3.1 with 2^{20} randomly chosen pairs (a, b) such that $a \in [1, 2^{609}), b \in [1, 2^{612})$ and $a, b \in [1, 2^{1221})$. The intervals $([1, 2^{609}), [1, 2^{612}))$ are relevant for obtaining a faster single-exponentiation algorithm at the 128-bit security level (see Section 3.3.1), and the interval $[1, 2^{1221})$ is relevant for deploying our double-exponentiation algorithm in the Nyberg-Rueppel signature scheme at the 128-bit security level (see Section 3.3.2). Our experimental evidence suggests that the main loop in steps 7–9 of Algorithm 3.1 is executed approximately $1.45 \log_2 (A + B)$ times on average (where $A = a/\gcd(a,b)$, $B = b/\gcd(a,b)$), and that the average number of multiplications per iteration is 4.39. Moreover, R1 is used in around 61% of the total number of iterations. In our experiments, as one might expect, $\gcd(a, b)$ is very small and the cost of step 10 is $2.32M_1$ on average (see Table 3.4). Note that step 11 can be performed at a negligible cost. Hence, we may conjecture that the expected running time of Algorithm 3.1 is $(1.45 \cdot (0.61 \cdot 4 + 0.39 \cdot 5)) \log_2 (a + b)M_1 \approx 6.37 \log_2 (a + b)M_1$.

3.3 Applications of double-exponentiation in factor-4 groups

In this section we discuss two applications of double-exponentiation of compressed elements in factor-4 groups. We show in Section 3.3.1 that double-exponentiation can be used to obtain faster single-exponentiation of compressed elements in factor-4 groups. Next, we show in Section 3.3.2 that double-exponentiation allows us to use compression techniques in ElGamal type signature schemes and furthermore obtain shorter public keys. As an illustrative example, we provide details for the Nyberg-Rueppel signature scheme.

Table 3.4: Practical behavior of Algorithm 3.1 at the 128-bit security level. 2^{20} pairs (a, b) were randomly chosen such that $a \in [1, 2^{609}), b \in [1, 2^{612})$ and $a, b \in [1, 2^{1221})$. $A = a/\gcd(a, b)$ and $B = b/\gcd(a, b)$.

	Average		Standard deviation	
	$a \in [1, 2^{609})$, $b \in [1, 2^{612})$	$(a, b) \in [1, 2^{1221})$	$a \in [1, 2^{609})$, $b \in [1, 2^{612})$	$(a, b) \in [1, 2^{1221})$
$\log_2(a + b)$	611.4	1221	1.058	0.816
$\log_2(A + B)$	611	1221	1.251	1.053
# of iterations (of steps 7–9)	$1.451 \cdot \log_2(A + B)$	$1.453 \cdot \log_2(a + b)$	0.018	0.012
# of multiplications per iteration (during steps 7–9)	4.390	4.390	0.027	0.019
# of multiplications in step 10	2.320	2.315	5.414	5.412
Rule	Average usage		Standard deviation	
R1	0.610	0.610	0.027	0.019
R2	0.175	0.175	0.013	0.009
R3	0.129	0.129	0.012	0.008
R4	0.085	0.085	0.013	0.009

Throughout this section we let $\mu_n = \langle g \rangle$ be the factor-4 subgroup of $\mathbb{F}_{q^4}^*$ with trace t, and let $T = |t|$.

3.3.1 Speeding up single-exponentiation

Algorithm 3.1 can be turned into a single-exponentiation algorithm as follows. Suppose we want to compute c_{rs} given c_r and $0 \leq s < n$. Clearly, if $s = 0$ then $c_{rs} = 0$. Suppose $s \neq 0$, and write $s = aT + b$ where $0 \leq a \leq T$ and $0 \leq b < T$ (this can be done as $T = \sqrt{2q}$ and $n = q + 1 - t$). If $a = 0$ then $c_{rs} = c_{rb}$ can be computed using Algorithm 2.1. If $b = 0$ then $c_{rs} = c_{raT} = c_{ra}^T$ (recall that T is a power of 2) can be computed by first computing c_{ra} from c_r and a using Algorithm 2.1, and then raising the result to the T'th power. Now, suppose $a, b \neq 0$. Then $c_{rs} = c_{arT+br} = c_{ak+bl}$, where $k = rT$ and $l = r$. In other words, c_{rs} can be computed from $a, b, c_l = c_r$ and $s_{k,l} = [c_{r(T-2)}, c_{r(T-1)}, c_{rT}, c_{r(T+1)}]$ using Algorithm 3.1. Note that c_l is already given, and $s_{k,l}$ is the last s_u value obtained from running Algorithm 2.1 with input c_r and T. Our discussion yields Algorithm 3.2 for single exponentiation in μ_n.

Let $C_1(i)$ denote the cost of Algorithm 2.1 when the input exponent b is approximately an $\lfloor i \rfloor$-bit integer. Similarly, let $C_2(i)$ denote the cost of Algorithm 3.1 when the sum of the two input exponents is approximately an $\lfloor i \rfloor$-bit integer. If $s \approx n$ and if one uses Algorithm 2.1 to compute $[c_{r(T-2)}, c_{r(T-1)}, c_{rT}, c_{r(T+1)}]$ in step 12 of Algorithm 3.2 then

Algorithm 3.2 A single-exponentiation algorithm
Input: c_r and s
Output: c_{rs}

1: **if** $s = 0$ **then**
2: $c_{rs} \leftarrow 0$
3: **else**
4: Write $s = aT + b$, where $T = |t|$, $0 \le a \le T$, $0 \le b < T$
5: **if** $a = 0$ **then**
6: Use Algorithm 2.1 to compute c_{rb} from c_r and b
7: $c_{rs} \leftarrow c_{rb}$
8: **else if** $b = 0$ **then**
9: Use Algorithm 2.1 to compute c_{ra} from c_r and a
10: $c_{rs} \leftarrow c_{ra}^T$
11: **else**
12: Compute $s_u = [c_{r(T-2)}, c_{r(T-1)}, c_{rT}, c_{r(T+1)}]$ from c_r and T
13: $s_{k,l} \leftarrow s_u,\ c_l \leftarrow c_r$
14: Use Algorithm 3.1 to compute c_{ak+bl} from a, b, c_l and $s_{k,l}$
15: $c_{rs} \leftarrow c_{ak+bl}$
16: **end if**
17: **end if**
18: Return c_{rs}

the running time of Algorithm 3.2 is approximately $C_1((\log_2 n)/2) + C_2((\log_2 n)/2)$ since $T \approx \sqrt{n}$. Therefore, we can conclude from Table 3.1, and from the running time analysis of Algorithm 3.1 at the end of Section 3.2, that the running time of Algorithm 3.2 will not exceed $(4\log_2 n + 12.4\log_2 n)M_1/2 = 8.2(\log_2 n)M_1$ and the running time of Algorithm 3.2 is expected to be $(4\log_2 n + 6.37\log_2 n)M_1/2 \approx 5.19(\log_2 n)M_1$. Moreover, in the case that the base c_r is fixed, $s_{k,l}$ in Algorithm 3.2 can be precomputed, and the running time of Algorithm 3.2 is expected to be $6.38(\log_2 n)M_1/2 = 3.19(\log_2 n)M_1$. Assuming the base c_r is fixed, we may conclude that Algorithm 3.2 is faster than Algorithm 2.1. However, for general bases, Algorithm 2.1 remains the fastest single exponentiation algorithm in factor-4 groups, unless we can find a more efficient method for computing $[c_{r(T-2)}, c_{r(T-1)}, c_{rT}, c_{r(T+1)}]$ given c_r and T.

We now argue that $[c_{r(T-2)}, c_{r(T-1)}, c_{rT}, c_{r(T+1)}]$ can be computed at a negligible cost. This will refine Algorithm 3.2 and give a faster single-exponentiation algorithm than Algorithm 2.1 for general bases. We first need the following theorem.

Theorem 3.3.1. *Let $\mu_n = \langle g \rangle \subset \mathbb{F}_{q^4}^*$ be a factor-4 group with trace t, and let $c_r = \mathrm{Tr}(g^r)$. Then*
(i) $c_{rt} = c_r^t$.

(ii) $c_{r(t-1)} = c_r$.
(iii) $c_{r(t-2)} = c_r^t$.
(iv) $c_{r(t+1)}$ *is a root of* $F(x) = x^2 + c_r^{t+1}x + (c_r^{t+2} + c_r^{3t} + c_r^2 + c_r^4)$.

Proof. (i) follows because t is a power of 2 and $\mathrm{char}(\mathbb{F}_q) = 2$.
(ii) $c_{r(t-1)} = c_{rq} = c_r^q = c_r$ since $q \equiv t - 1 \pmod{n}$.
(iii) $c_{r(t-2)} = c_{rqt} = c_r^t$ since $qt \equiv q^2 + q \equiv q - 1 \equiv t - 2 \pmod{n}$.
(iv) First note that

$$\begin{aligned} c_{r(2t-1)} = \quad & \frac{1}{c_r^{t+1}}\left((c_{r(t+1)} + c_{rt} + c_{r(t-1)} + c_{r(t-2)})^2\right) \\ & + \frac{1}{c_r^{t+1}}\left((c_{rt} + c_{r(t-1)})^2(c_r^t + c_{rt}^2)\right), \end{aligned} \tag{3.3.1}$$

by using Theorem 3.1.4(iii) with $u = rt$ and $v = r$. Moreover, using Lemma 3.1.2(iv) with $u = rt$ and $v = r$ together with the fact that $c_{r(t-1)} = c_r$ from part (ii), we can show that

$$c_{r(2t-1)} = c_r^{t+1} + c_{r(t+1)}. \tag{3.3.2}$$

Now, combining (3.3.1) and (3.3.2) we can write

$$c_{r(t+1)}^2 + c_r^{t+1}c_{r(t+1)} + (c_r^{t+2} + c_r^{3t} + c_r^2 + c_r^4) = 0,$$

which proves the result. □

Suppose now that c_r is known and that $t > 0$, whence $T = t$ (we can similarly argue when $t < 0$). By Theorem 3.3.1(i), (ii), and (iii), we can compute $[c_{r(T-2)}, c_{r(T-1)}, c_{rT}]$ at a cost of $2F_1$. We also know that the roots of $F(x) = x^2+Bx+C$, $B, C \in \mathbb{F}_q$, $B \neq 0$, are given by $\{r_1, r_2\} = \{B\cdot R(C/B^2), B\cdot(R(C/B^2)+1)\}$, where $R(C/B^2)$ is a root of $x^2+x+(C/B^2)$, and if $r_1, r_2 \in \mathbb{F}_q$ then they can be computed at a negligible cost (see [40, Section 3.6.2]). By Theorem 3.3.1(iv), the roots of $F(x) = x^2 + c_r^{t+1}x + (c_r^{t+2} + c_r^{3t} + c_r^2 + c_r^4)$ are in $\mathbb{F}_q$, and so can be computed efficiently. Hence, we can determine $[c_{r(T-2)}, c_{r(T-1)}, c_{rT}, c_{r(T+1)}]$ at a negligible cost, up to the ambiguity that we cannot differentiate between the roots $c_{r(T+1)}$ and $c_{r(T+1)} + c_r^{T+1}$ of $F(x)$ (see Theorem 3.3.1). This ambiguity problem can be resolved if the sender of c_r is also required to compute $c_{r(T+1)}$ and send one extra bit $b \in \{0, 1\}$ to help the receiver distinguish $c_{r(T+1)}$ from $c_{r(T+1)} + c_r^{T+1}$ (see Section 3.3.2 for a similar discussion on how b can be determined).

Hence, assuming that we have a general base c_r and that the distinguisher bit b is known, the running time of Algorithm 3.2 does not exceed $(12.4\log_2 n)M_1/2 = 6.2(\log_2 n)M_1$. Based on our experiments (see Table 3.4), the running time of Algorithm 3.2 is expected to be $(6.37\log_2 n)M_1/2 \approx 3.19(\log_2 n)M_1$; this is 20% faster than Algorithm 2.1 which

requires a negligible precomputation and has running time $4(\log_2 n)M_1$ (see Table 3.1). Note that the Diffie-Hellman key exchange protocol is an example of the scenario where c_r and b can be computed from c_1 by one of the communicating parties and sent to the other party.

To be more concrete, we compare the expected running time of our new single-exponentiation algorithm (Algorithm 3.2) with the previously known fastest single-exponentiation algorithm (Algorithm 2.1) at the 128-bit security level when general bases are employed. We let $q = 2^{1223}$ and $t = 2^{612}$. Then $q + 1 - t = 5n$ where n is a 1221-bit prime, and $\mathbb{F}_{q^4}^*$ has a factor-4 subgroup μ_n of order n. For any exponent $s \in [1, n-1]$, we can write $s = a \cdot 2^{612} + b$ where a has bitlength at most 609 and b has bitlength at most 612, and compute c_{rs} using Algorithm 3.2 with input c_r, s and these a, b values. Based on our experiments, the average cost of Algorithm 3.2 is $3895M_1$ (see Table 3.4). Note that this is 20% less than the cost of Algorithm 2.1, which is $4880M_1$.

3.3.2 Nyberg-Rueppel signature scheme in factor-4 groups

The Nyberg-Rueppel signature scheme [72] can be modified to be used with compressed representations of elements in μ_n as follows. We suppose that q, n and c_1 are system-wide parameters; alternately, they may be included in a party's public key. Alice chooses a random integer $k \in [1, n-1]$ and computes $s_{k,1} = [c_{k-2}, c_{k-1}, c_k, c_{k+1}]$. Alice's public key is $s_{k,1}$ and her private key is k. To sign a message M, Alice chooses a random integer $a \in [1, n-1]$ and computes c_a. Alice extracts a session key $K = \mathrm{Ext}(c_a)$ from c_a, and uses a symmetric-key encryption function E to encipher M under K: $e = \mathrm{E}_K(M)$. Moreover, she computes the hash $h = \mathrm{H}(e)$ of the encrypted text and sets $s = k \cdot h + a \bmod n$. Alice's signature on M is (e, s).

If Bob wants to verify Alice's signature (e, s) on M, he first computes $h = \mathrm{H}(e)$, replaces h by $-h \bmod n$, and computes c_{hk+s} from $s_{k,1}$ and c_1. Note that this is a double-exponentiation to the base $(k, 1)$. Bob extracts the session key $K' = \mathrm{Ext}(c_{hk+s})$ from c_{hk+s} and computes $e' = \mathrm{E}_{K'}(M)$. Bob accepts the signature if and only if $e' = e$.

One advantage of using the compressed representation of elements in μ_n with the Nyberg-Rueppel signature scheme is that c_1 is a system parameter instead of (the longer) g. We further show that it is possible to have a shorter public key at the expense of some negligible precomputation. In particular, we show that c_{k+1} is a root of a certain quadratic polynomial P_k whose coefficients are determined by c_1, c_{k-1} and c_k. That is, Alice can omit c_{k+1} from her public key and instead specify one bit to help Bob distinguish c_{k+1} from the other root of P_k.

Consider the matrix

$$M_u = \begin{pmatrix} c_u & c_{u+1} & c_u & c_{u+1} \\ c_{u+1} & c_{u+2} & c_{u+3} & c_{u+4} \\ c_{u+2} & c_{u+3} & c_{u+4} & c_{u+5} \\ c_{u+3} & c_{u+4} & c_{u+5} & c_{u+6} \end{pmatrix}.$$

Giuliani and Gong [35] showed that the characteristic polynomial of the matrix $M_u^{-1} \cdot M_{u+k}$ is equal to the characteristic polynomial of g^k over $\mathbb{F}_q$, namely $f_{g^k}(x) = x^4 + c_k x^3 + c_k^t x^2 + 1$. In particular, using Lemma 3.1.2(i), we can compute the characteristic polynomial P_k of $M_{-2}^{-1} \cdot M_{k-2}$ and find that the coefficient $C_2(P_k)$ of x^2 in $P_k(x)$ satisfies

$$\begin{aligned} (c_1 c_t)^2 C_2(P_k) &= (c_1^2 + c_t^2)c_{k+1}^2 + (c_1 c_t((c_{k-2} + c_k) + c_1 c_{k-1} + c_k c_t))c_{k+1} \\ &\quad + (c_1(c_{k-2} + c_k))^2 + c_1^2 c_t(c_{k-2}c_k + (c_{k-1} + c_k)^2) \\ &\quad + c_1 c_{k-1} c_t(c_1^2 c_k + c_{k-2} + c_k) + (c_{k-1}(c_1 + c_1^2 + c_t))^2 \\ &\quad + c_k^2(c_1^4 + c_t^3). \end{aligned}$$

We should note that $c_1 \neq 0$ as otherwise f_g would not be irreducible over $\mathbb{F}_q$. Since $P_k = f_{g^k}$, we must have $C_2(P_k) = c_k^t$ yielding the following result.

Theorem 3.3.2. *c_{k+1} is a root of the polynomial $P_k(x) = Ax^2 + Bx + C$ where*

$$\begin{aligned} A &= c_1^2 + c_t^2 \\ B &= c_1 c_t((c_{k-2} + c_k) + c_1 c_{k-1} + c_k c_t) \\ C &= (c_1(c_{k-2} + c_k))^2 + c_1^2 c_t(c_{k-2}c_k + (c_{k-1} + c_k)^2 + c_k^t c_t) \\ &\quad + c_1 c_{k-1} c_t(c_1^2 c_k + c_{k-2} + c_k) + (c_{k-1}(c_1 + c_1^2 + c_t))^2 \\ &\quad + c_k^2(c_1^4 + c_t^3). \end{aligned}$$

First note that $A = 0$ if and only if $c_1 = c_t$, that is, if and only if g and g^t are conjugates over $\mathbb{F}_q$. Using $q^2 \equiv -1 \pmod{n}$ and $q \equiv t - 1 \pmod{n}$, we can show that this is never the case.

Now, the roots of $P_k(x) = Ax^2 + Bx + C$, $B, C \in \mathbb{F}_q$, $A, B \neq 0$, are given by $\{r_1, r_2\} = \{(B/A) \cdot R(AC/B^2), (B/A) \cdot (R(AC/B^2) + 1)\}$, where $R(AC/B^2)$ is a root of $x^2 + x + (AC/B^2)$. Furthermore, if $r_1, r_2 \in \mathbb{F}_q$ then they can be computed at a negligible cost (see [40, Section 3.6.2]).

If $A, B \neq 0$, then Alice's public key can be $([c_{k-2}, c_{k-1}, c_k], b)$ where $b \in \{0, 1\}$ is determined by Alice to help Bob to distinguish c_{k+1} from the other root of P_k. For example, Alice can do this as follows. She first computes c_{k+1} (c_{k+1} is obtained for free while computing c_k from c_1 using a single-exponentiation algorithm) and determines the roots

r_1, r_2 of P_k together with integers corresponding to the bit representations of r_1 and r_2, say i_1 and i_2, respectively. We may assume without loss of generality that $i_1 < i_2$. If $c_{k+1} = r_1$ then Alice sets $b = 0$, otherwise she sets $b = 1$. Given Alice's public key, Bob can easily recover $s_{k,1}$ at a negligible cost and verify Alice's signatures.

If $A \neq 0$ and $B = 0$ in $P_k(x)$, then c_{k+1} can be uniquely computed by taking the square root of C/A in $\mathbb{F}_q$ at a negligible cost.

To be more concrete, we compare the length of public keys when compression is deployed with the length of public keys in the original scheme at the 128-bit security level. As in Section 3.3.1, we let $q = 2^{1223}$ and $t = 2^{612}$, whence $q + 1 - t = 5n$ where n is a 1221-bit prime and $\mathbb{F}_{q^4}^*$ has a factor-4 subgroup μ_n of order n. We find that the length of a public key when compression is deployed is 3669 bits, while the public key length without compression is 4892 bits.

3.4 Concluding remarks

We showed that double-exponentiation can be efficiently performed using factor-4 compressed representation of elements in factor-4 groups. This allowed us to speed up single-exponentiation and also to use compression techniques in ElGamal type signature schemes.

A future research goal is to further speed up single-exponentiation and double-exponentiation algorithms that use compressed representation of elements. One possibility is that Algorithm 3.1 can be optimized by taking into account update rules different from those given in Table 3.2 (see for example [68] and [84]). Another possibility is to search for better parameters. For example, at the 128-bit security level, generating parameters q, n such that $\mu_n \subset \mathbb{F}_{q^4}^*$ is a factor-4 group, $q = 2^m \approx 2^{1200}$, and n is a 256-bit prime, would result in shorter exponents and therefore a speed up by a factor of more than 4. Such parameters can be found by searching for a suitable prime factor n of $N_1 = \gcd(\Phi_{4m}(2), q + 1 - t)$ or $N_2 = \gcd(\Phi_{4m}(2), q + 1 + t)$, where Φ_{4m} is the $(4m)$th-cyclotomic polynomial of degree $\varphi(4m)$, and φ is Euler's totient function. Note that when $\varphi(m)$ is significantly smaller than m, then factoring N_i is expected to be easier than factoring $q + 1 \pm t$. For example, when $m = 1209$, N_2 is a 718-bit integer and has a 271-bit prime factor n.

We expect that our double-exponentiation algorithm can be adapted to the factor-6 groups that arise as multiplicative subgroups of characteristic-three finite fields [78, 50]. However, further analysis would be needed to estimate its efficiency, and to judge how the resulting single-exponentiation method compares to previously known algorithms.

Chapter 4

Torus-Based Compression by a Factor-4 and 6

In this chapter, we extend the torus-based compression technique for cyclotomic subgroups $G_{q,4} \subseteq \mathbb{F}_{q^4}^*$ and $G_{q,6} \subseteq \mathbb{F}_{q^6}^*$, and obtain compression factor-4 for the subgroups $G \subsetneq G_{q,4}$ of orders $q \pm \sqrt{2q} + 1$, where $q = 2^m$, m is odd, and factor-6 compression for the subgroups $G \subsetneq G_{q,6}$ of orders $q \pm \sqrt{3q} + 1$, where $q = 3^m$, m is odd. We present exponentiation algorithms that use torus-based compressed representations of elements, and compare them to the trace-based exponentiation algorithms. We also argue that, in general, the torus-based compression technique is unlikely to extend to obtain better compression factors for proper subgroups of cyclotomic subgroups of $G_{q,k}$.

The remainder of this chapter is organized as follows. In Section 4.1, we give some details on torus-based compression. In Section 4.2, we present our argument for the difficulty of obtaining the *optimal* compression factor k for the elements of $G_\ell \subsetneq G_{q,k} \subseteq \mathbb{F}_{q^k}$, where $\ell \approx q$. We analyze two particular cases in Sections 4.3 and 4.4 and show that, in contrast to our pessimistic arguments in Section 4.2, one can obtain factor-4 and factor-6 compression by extending torus-based compression techniques. In Sections 4.5 and 4.6 we describe several exponentiation algorithms based on our compression/decompression techniques. In Section 4.7 we give a comparison of exponentiation algorithms and conclude that torus-based exponentiation algorithms outperform the trace-based exponentiation algorithms and the conventional exponentiation algorithms. We state some open problems in Section 4.8.

4.1 A review of torus-based compression

Let q be a prime power and $\mathbb{F}_q$ denote the finite field of order q, and recall that $G_{q,k}$ is the cyclotomic subgroup of $\mathbb{F}_{q^k}^*$, where $|G_{q,k}| = \Phi_k(q)$. Throughout this chapter, we will denote the trace function $\mathrm{Tr}_{\mathbb{F}_{q^i}/\mathbb{F}_{q^j}} : \mathbb{F}_{q^i} \to \mathbb{F}_{q^j}$ by Tr_{q^i,q^j}.

Let V be an algebraic variety defined over $\mathbb{F}_{q^k}$ via the set of solutions of a system of (multivariate) polynomial equations $(f_1(X), f_2(X), \ldots, f_r(X))$, $f_i \in \mathbb{F}_{q^k}[X]$. Note that $(f_1(X), f_2(X), \ldots, f_r(X))$ can equivalently be represented as a system of polynomials $(g_1(Y), g_2(Y), \ldots g_s(Y))$ defined over $\mathbb{F}_q$, by considering $\mathbb{F}_{q^k}$ as a k-dimensional vector space over $\mathbb{F}_q$. The algebraic variety that corresponds to this new system of polynomial equations is called the *Weil restriction* of scalars from $\mathbb{F}_{q^k}$ to $\mathbb{F}_q$ and denoted $\mathrm{Res}_{\mathbb{F}_{q^k}/\mathbb{F}_q} V$. In particular, $(\mathrm{Res}_{\mathbb{F}_{q^k}/\mathbb{F}_q} V)(\mathbb{F}_q) \cong V(\mathbb{F}_{q^k})$, where $V(\mathbb{F})$ denotes the $\mathbb{F}$-points of V. For example, if $\mathbb{G}_m$ is the multiplicative group then we have $(\mathrm{Res}_{\mathbb{F}_{q^k}/\mathbb{F}_q} \mathbb{G}_m)(\mathbb{F}_q) \cong \mathbb{G}_m(\mathbb{F}_{q^k}) = \mathbb{F}_{q^k}^*$.

Definition 4.1.1. [74] An *algebraic torus* $\mathbb{T}$ of dimension d over $\mathbb{F}_q$ is an algebraic group defined over $\mathbb{F}_q$ that, over some finite extension field $\mathbb{F}$, is isomorphic to $(\mathbb{G}_m)^d$. Such a field $\mathbb{F}$ is called a splitting field for $\mathbb{T}$; or we say $\mathbb{T}$ splits over $\mathbb{F}$.

Definition 4.1.2. [74] A *rational map* between algebraic varieties is a function defined by quotients of polynomials that is defined almost everywhere. A *birational isomorphism* between algebraic varieties is a rational map that has a rational inverse. A d-dimensional variety over $\mathbb{F}_q$ is *rational* over $\mathbb{F}_q$ if it is birationally isomorphic over $\mathbb{F}_q$ to $\mathbb{A}^d$, where $\mathbb{A}$ is the affine d-space. A variety V over $\mathbb{F}_q$ is called *stably rational* over $\mathbb{F}_q$ if $V \times \mathbb{A}^r$ is rational over $\mathbb{F}_q$ for some $r \geq 0$.

As noted in [74], $\mathrm{Res}_{\mathbb{F}_{q^k}/\mathbb{F}_q} \mathbb{G}_m$ is a k-dimensional algebraic torus over $\mathbb{F}_q$ that splits over $\mathbb{F}_{q^k}$. Rubin and Silverberg define a subvariety (and an algebraic subgroup) $\mathbb{T}_{q,k}$ of $\mathrm{Res}_{\mathbb{F}_{q^k}/\mathbb{F}_q} \mathbb{G}_m$ to be the intersection of the kernels of the norm maps $\mathbb{N}_{\mathbb{F}_{q^k}/\mathbb{F}} : \mathrm{Res}_{\mathbb{F}_{q^k}/\mathbb{F}_q} \mathbb{G}_m \to \mathrm{Res}_{\mathbb{F}/\mathbb{F}_q} \mathbb{G}_m$, for all $\mathbb{F}_q \subseteq \mathbb{F} \subsetneq \mathbb{F}_{q^k}$. That is,

$$\mathbb{T}_{q,k} = \mathrm{Ker}\left[\mathrm{Res}_{\mathbb{F}_{q^k}/\mathbb{F}_q} \mathbb{G}_m \xrightarrow{\oplus \mathbb{N}_{\mathbb{F}_{q^k}/\mathbb{F}}} \bigoplus_{\mathbb{F}_q \subseteq \mathbb{F} \subsetneq \mathbb{F}_{q^k}} \mathrm{Res}_{\mathbb{F}/\mathbb{F}_q} \mathbb{G}_m \right].$$

They show that $\mathbb{T}_{q,k}$ is an algebraic torus of dimension $\varphi(k)$ that splits over $\mathbb{F}_{q^k}$, and that

$$\mathbb{T}_{q,k}(\mathbb{F}_q) \cong \{g \in \mathbb{F}_{q^k} : \mathbb{N}_{\mathbb{F}_{q^k}/\mathbb{F}}(g) = 1 \text{ for all } \mathbb{F}_q \subseteq \mathbb{F} \subsetneq \mathbb{F}_{q^k}\} = G_{q,k}.$$

It is known that the torus $\mathbb{T}_{q,k}$ is rational over $\mathbb{F}_q$ if k is a prime power or a product of two prime powers; and that $\mathbb{T}_{q,k}$ is always stably rational. In particular, Rubin and

Silverberg presented explicit birational maps (or equivalently compression/decompression maps) $\rho : \mathbb{T}_{q,k} \to \mathbb{A}^{\varphi(k)}$ when $k = 2$ and $k = 6$, thereby achieving compression factors $2/\varphi(2) = 2$ and $3 = 6/\varphi(6)$ for groups $G_{q,k}$. The compression and decompression maps that correspond to the case $k = 2$ will be the building blocks in our arguments, so we explicitly state them here for future reference.

Let $\mathbb{F}_{q^2} = \mathbb{F}_q[\sigma]/(f(\sigma))$. If q is even, we set $f(\sigma) = \sigma^2 + \sigma + c$ with $c \in \mathbb{F}_q$ and $\mathrm{Tr}_{q^2,2}(c) = 1$. If q is odd, we set $f(\sigma) = \sigma^2 - c$ where $c \in \mathbb{F}_q$ is a quadratic non-residue. Then

$$\begin{aligned} \mathcal{C} : \ G_{q,2} \setminus \{\pm 1\} &\to \mathbb{F}_q \\ g_0 + g_1\sigma &\mapsto \frac{g_0+1}{g_1}, \end{aligned} \tag{4.1.1}$$

and

$$\begin{aligned} \mathcal{D} : \ \mathbb{F}_q &\to G_{q,2} \\ \alpha &\mapsto \begin{cases} \frac{\alpha+\sigma}{\alpha+1+\sigma} & \text{if } q \text{ is even,} \\ \frac{\alpha+\sigma}{\alpha-\sigma} & \text{if } q \text{ is odd,} \end{cases} \end{aligned} \tag{4.1.2}$$

define the compression and the decompression maps, respectively [74]. After observing that

$$\begin{aligned} G_{q,2} &= \{g_0 + g_1\sigma : \ g_0, g_1 \in \mathbb{F}_q \text{ and } (g_0 + g_1\sigma)^{q+1} = 1\} \\ &= \begin{cases} \{g_0 + g_1\sigma : \ g_0, g_1 \in \mathbb{F}_q \text{ and } g_0^2 + cg_1^2 + g_0g_1 = 1\} & \text{if } q \text{ is even,} \\ \{g_0 + g_1\sigma : \ g_0, g_1 \in \mathbb{F}_q \text{ and } g_0^2 - cg_1^2 = 1\} & \text{if } q \text{ is odd,} \end{cases} \end{aligned}$$

one can check that $\mathcal{C}$ and $\mathcal{D}$ are inverses of each other when they are defined, and that

$$\mathcal{D}(\alpha)\mathcal{D}(\beta) = \mathcal{D}\left(\frac{\alpha\beta + c}{\alpha + \beta + 1}\right) \quad \text{if } q \text{ is even,} \tag{4.1.3}$$

$$\mathcal{D}(\alpha)\mathcal{D}(\beta) = \mathcal{D}\left(\frac{\alpha\beta + c}{\alpha + \beta}\right) \quad \text{if } q \text{ is odd.} \tag{4.1.4}$$

We note that formulas (4.1.3) and (4.1.4) can be used to perform multiplication and exponentiation in $G_{q,2} \setminus \{\pm 1\}$ when working with the compressed representation of elements in $\mathbb{F}_q$.

4.2 On the (im)possibility of optimal compression

In Section 4.1 we saw that one can at best hope to compress the elements of $G_{q,k} \subset \mathbb{F}_{q^k}^*$ by a factor $k/\varphi(k)$ which seems to be the optimal compression factor as $|G_{q,k}| \approx q^{\varphi(k)}$.

However, we also saw in Chapter 2 that for $k = 4$ and $k = 6$, one can compress further by a factor of 2 and obtain compression factor $2k/\varphi(k) = k$ for the elements of certain proper subgroups G_ℓ of $G_{q,k}$. In particular, for $k = 4$ we have $q = 2^m$ and $\ell = q \pm \sqrt{2q} + 1$, and for $k = 6$ we have $q = 3^m$ and $\ell = q \pm \sqrt{3q} + 1$; in both cases, m is odd. Note that, in both cases, $G_\ell \approx q$ and so k is the optimal compression factor. In general, it would be desirable to compress the elements of any order-ℓ subgroup $G_\ell \subsetneq G_{q,k} \subseteq \mathbb{F}_{q^k}^*$ by an *optimal* factor $(k \log q)/\log \ell$ in any characteristic.

We first recall some details on how to achieve compression factor $k = 4$ in characteristic-two fields using the trace representation of elements, and explain why this compression technique does not seem to generalize to fields with characteristic different from two.

Let $q = 2^m, t = \sqrt{2q}$ and $\ell = q \pm t + 1$, where m is odd. We note that $\Phi_4(q) = q^2 + 1 = (q + t - 1)(q - t + 1)$ and $G_\ell \subset G_{q,4} \subset \mathbb{F}_{q^4}^*$ has embedding degree $k = 4$ with respect to q. In Chapter 2, we have shown that if $g \in G_\ell$ then the minimal polynomial f_g of g over $\mathbb{F}_q$ is

$$f_g(x) = x^4 - \mathrm{Tr}_{q^4,q}(g)x^3 + \mathrm{Tr}_{q^4,q}(g)^t x^2 - \mathrm{Tr}_{q^4,q}(g)x + 1.$$

One readily deduces that g can be uniquely identified up to conjugation over $\mathbb{F}_q$ from $\mathrm{Tr}_{q^4,q}(g)$, thereby achieving factor-4 compression. A natural extension is to try to represent the elements of G_ℓ with embedding degree 4 using their traces over $\mathbb{F}_q$, where q is not even. We fix parameters (q, ℓ) such that q is a prime power, $\gcd(q, \ell) = 1$, $q^2+1 \equiv 0 \pmod{\ell}$, and $q^i \not\equiv 1 \pmod{\ell}$ for $1 \leq i < 4$. We have shown in Chapter 2 that the minimal polynomial f_g of $g \in G_\ell$ over $\mathbb{F}_q$ can be computed as

$$f_g(x) = x^4 - \mathrm{Tr}_{q^4,q}(g)x^3 + (\mathrm{Tr}_{q^4,q}(g^{q+1}) + 2)x^2 - \mathrm{Tr}_{q^4,q}(g)x + 1.$$

Therefore, two $\mathbb{F}_q$-elements (as opposed to only one) are generally required to identify g uniquely up to its conjugates over $\mathbb{F}_q$, unless one of $\mathrm{Tr}_{q^4,q}(g)$ or $\mathrm{Tr}_{q^4,q}(g^{q+1})$ can be obtained from the other. In fact, one can find parameters (q, ℓ), and elements $g_1, g_2 \in G_\ell \subsetneq G_{q,4}$ such that g_1 and g_2 are not conjugates over $\mathbb{F}_q$ but $\mathrm{Tr}_{q^4,q}(g_1) = \mathrm{Tr}_{q^4,q}(g_2)$.

Next, we give some evidence that, in general, compressing the elements of $G_\ell \subsetneq G_{q,k} \subseteq \mathbb{F}_{q^k}^*$ by an optimal factor $(k \log q)/\log \ell$ might not be possible using *tori-like* techniques. Again, we consider the case $k = 4$ and $\ell \approx q$. Note that $(k \log q)/\log \ell \approx 4$. Let $\mathbb{F}_{q^2} = \mathbb{F}_q[w]/(g(w))$ and $\mathbb{F}_{q^4} = \mathbb{F}_{q^2}[\sigma]/(f(\sigma))$ for some suitable f and g. Let $g = g_0 + g_1\sigma \in G_\ell$ and recall from Section 4.1 that if $g \neq \pm 1$ then it can be uniquely identified with an element $\alpha \in \mathbb{F}_{q^2}$ if $f(\sigma)$ is of the form $\sigma^2 - c$ or $\sigma^2 + \sigma + c$; see (4.1.1) and (4.1.2). More precisely, if $g = g_0 + g_1\sigma$ then $\alpha = (g_0 + 1)/g_1$ and $g = \mathcal{D}(\alpha)$, where $\mathcal{D}(\alpha) = (\alpha + \sigma)/(\alpha - \sigma)$ if q is even, and $\mathcal{D}(\alpha) = (\alpha + \sigma)/(\alpha + 1 + \sigma)$ if q is odd. Let $\alpha = a + bw$ for some $a, b \in \mathbb{F}_q$. Note that the compression of g into (a, b) does not utilize the fact that g lies in a proper subgroup G_ℓ of $G_{q,4}$. Therefore, one might try to compress g further into b (or a), by using the relation $g^\ell = 1$ to obtain an expression for one of a and b in terms of the other. For

example, the most naive way would be to use the relation $g^\ell = 1$ and obtain a polynomial $P(x, y) \in \mathbb{F}_q[x, y]$ such that $P(a, b) = 0$. Then we would hope to find a among the roots of $P(x, b) = 0$. Since we also want the corresponding decompression function to be efficiently computable and *almost* one-to-one we might ask the following question.

Question: What is the minimum expected degree and sparsity of a polynomial $P(x, y) \in \bar{\mathbb{F}}_q[x, y]$ such that (i) for (*almost*) all $b \in \mathbb{F}_q$ there exists an (*almost*) unique solution $a \in \mathbb{F}_q$ to $P(x, b) = 0$; and (ii) for $\alpha = a + bw$ we have $\mathcal{D}(\alpha) \in G_\ell$.

Remark 4.2.1. Given $\binom{n+m}{n}$ pairs $(X_i, z_i) \in \mathbb{F}_q^m \times \mathbb{F}_q$, an nth degree polynomial P in m variables can be constructed such that $P(X_i) = z_i$. Note that $\binom{n+m}{n}$ is the number of ways of choosing n elements from a set of $m + 1$ elements with repetitions allowed, which is therefore the maximum number of monomials in P. Hence, when $m = 2$ we would expect $\deg(P(x, y)) \leq \sqrt{\ell} \approx \sqrt{q}$.

4.3 Factor-4 compression in characteristic two

Let $q = 2^m$, m odd, $t = \sqrt{2q}$, $\ell = q + 1 - t$ and $\bar{\ell} = q + 1 + t$. Then

$$\begin{aligned} q^4 - 1 &= (q^2 - 1)(q^2 + 1) \\ &= (q^2 - 1)(q + 1 - t)(q + 1 + t). \end{aligned}$$

Let $G_\ell \subset G_{q,4} \subset \mathbb{F}_{q^4}^*$ and $G_{\bar{\ell}} \subset G_{q,4} \subset \mathbb{F}_{q^4}^*$ be subgroups such that $|G_{q,4}| = q^2 + 1$, $|G_\ell| = \ell$ and $|G_{\bar{\ell}}| = \bar{\ell}$. In this section, we set $\mathbb{F}_{q^2} = \mathbb{F}_q[w]/(w^2 + w + c_0)$ and $\mathbb{F}_{q^4} = \mathbb{F}_{q^2}[\sigma]/(\sigma^2 + \sigma + c_1)$. We must have $\mathrm{Tr}_{q,2}(c_0) = \mathrm{Tr}_{q^2,2}(c_1) = 1$.

Lemma 4.3.1. *Let* $\mathbb{F}_{q^2} = \mathbb{F}_q[w]/(w^2 + w + c_0)$ *and* $\mathbb{F}_{q^4} = \mathbb{F}_{q^2}[\sigma]/(\sigma^2 + \sigma + c_1)$ *with* $\mathrm{Tr}_{q,2}(c_0) = \mathrm{Tr}_{q^2,2}(c_1) = 1$. *Then*

$$\sigma^q + \sigma = \sum_{i=0}^{m-1} c_1^{2^i} = u_0 + u_1 w, \tag{4.3.1}$$

$$\sigma^t + \sigma = \sum_{i=0}^{(m-1)/2} c_1^{2^i} = u_2 + u_3 w, \tag{4.3.2}$$

$$w^q + w = 1, \tag{4.3.3}$$

$$w^t + w = \sum_{i=0}^{(m-1)/2} c_0^{2^i} = u_4, \tag{4.3.4}$$

for some $u_i \in \mathbb{F}_q$. *In particular,* $u_1 = 1$.

Proof. The equalities can be proven by repeatedly squaring the equations $\sigma^2 + \sigma = c_1$ and $w^2 + w = c_0$. We have $u_1 = 1$ since

$$1 = \mathrm{Tr}_{q^2,2}(c_1) = (\sigma + \sigma^q) + (\sigma + \sigma^q)^q = u_1(w + w^q) = u_1.$$

□

We furthermore assume throughout this section that $\sigma^2 + \sigma = c_1 = u_5 + u_6 w$, where $u_5, u_6 \in \mathbb{F}_q$.

Let $g = g_0 + g_1\sigma \in G_{q,4}$. We already know from Section 4.1 that if $g \neq 1$ then g can be compressed to an element $\alpha = (g_0 + 1)/g_1 \in \mathbb{F}_{q^2}$, and that a compressed element $\alpha \in \mathbb{F}_{q^2}$ can be decompressed to obtain $g = (\alpha + \sigma)/(\alpha + 1 + \sigma) \in G_{q,4} \setminus \{1\}$. Our objective is to show that $g \in \{G_\ell, G_{\bar{\ell}}\} \setminus \{1\}$ can further be compressed to $b \in \mathbb{F}_q$, and that a compressed $b \in \mathbb{F}_q$ can be decompressed to obtain $g \in \{G_\ell, G_{\bar{\ell}}\} \setminus \{1\}$. The following theorem plays a key role.

Theorem 4.3.2. *Let $g = (\alpha + \sigma)/(\alpha + 1 + \sigma) \in G_{q,4} \setminus \{1\}$ where $\alpha = a + bw \in \mathbb{F}_{q^2}$ for some $a, b \in \mathbb{F}_q$. If $g \in G_\ell$ then a is a root of the polynomial*

$$P_1(x, b) = x^t + x + b^{t+1} + (u_0 + u_4)b^t + (u_0 + u_3 + 1)b + (u_0u_3 + u_2 + u_6),$$

where the u_i's are as specified in Lemma 4.3.1. If $g \in G_{\bar{\ell}}$ then a is a root of the polynomial

$$P_1(x, b) = x^t + x + b^{t+1} + (u_0 + u_4)b^t + (u_0 + u_3 + 1)b + (u_0u_3 + u_2 + u_6 + 1).$$

Proof. Let $g = (\alpha+\sigma)/(\alpha+1+\sigma) \in G_\ell \setminus \{1\}$. After expanding and simplifying $g^{q+1-t} = 1$, we find that $\alpha + \sigma$ is a root of

$$P(x) = x^{q+t} + x^{q+1} + x^{t+1} + x^t.$$

Now, writing $\alpha = a + bw$ for some $a, b \in \mathbb{F}_q$ and simplifying $P(\alpha + \sigma) = 0$ gives us

$$\begin{aligned}
P(\alpha) &= a^2 + ab + (\sigma^q + \sigma)a + (w^2 + w)b^2 + ba^t + w^t b^{t+1} \\
&\quad + (w(\sigma^q + \sigma) + \sigma^t + \sigma)b + (\sigma^q + \sigma + 1)a^t \\
&\quad + (w^t(\sigma^q + \sigma + 1))b^t + (\sigma^q(\sigma^t + \sigma) + \sigma^t(\sigma + 1)) \\
&= a^2 + ab + (u_0 + w)a + c_0b^2 + ba^t + (u_4 + w)b^{t+1} \\
&\quad + ((u_0 + u_3 + 1)w + (c_0 + u_2))b + (u_0 + 1 + w)a^t \\
&\quad + ((u_0 + u_4)w + (c_0 + u_0u_4 + u_4))b^t + (u_0u_3 + u_2 + u_6)w \\
&\quad + (c_0u_3 + u_0u_2 + u_2 + u_5) \\
&= P_0(a, b) + P_1(a, b)w = 0,
\end{aligned}$$

where

$$\begin{aligned} P_0(a,b) &= a^t b + (u_0+1)a^t + a^2 + ab + u_0 a + u_4 b^{t+1} \\ &\quad +(c_0 + u_0u_4 + u_4)b^t + c_0 b^2 + (c_0 + u_2)b \\ &\quad +(c_0u_3 + u_0u_2 + u_2 + u_5), \\ P_1(a,b) &= a^t + a + b^{t+1} + (u_0+u_4)b^t + (u_0+u_3+1)b \\ &\quad +(u_0u_3 + u_2 + u_6). \end{aligned}$$

Hence, if $g = \frac{\alpha+\sigma}{\alpha+1+\sigma} \in G_\ell$ for some $\alpha = a + bw \in \mathbb{F}_{q^2}$ with $a, b \in \mathbb{F}_q$, we must have $P_0(a,b) = P_1(a,b) = 0$. In particular, a must be a root of the polynomial

$$P_1(x) = P_1(x,b) = x^t + x + b^{t+1} + (u_0+u_4)b^t + (u_0+u_3+1)b + (u_0u_3 + u_2 + u_6).$$

The case when $g \in G_{\bar{\ell}} \setminus \{1\}$ can be proved similarly. □

Lemma 4.3.3. *Let $P_1(x) = x^t + x + u \in \mathbb{F}_q[x]$. Then $P_1(x) = 0$ has a solution in $\mathbb{F}_q$ if and only if $\mathrm{Tr}_{q,2}(u) = 0$. If $\mathrm{Tr}_{q,2}(u) = 0$ then $P_1(x) = 0$ has exactly two solutions a_0, a_1 in $\mathbb{F}_q$, and $a_1 = a_0 + 1$.*

Proof. We first prove that $P_1(x) = 0$ has a solution in $\mathbb{F}_q$ if and only if $\mathrm{Tr}_{q,2}(u) = 0$. Suppose that $P_1(x) = x^t + x + u = 0$ has a solution, say $a \in \mathbb{F}_q$. Then

$$\mathrm{Tr}_{q,2}(u) = \mathrm{Tr}_{q,2}(a^t + a) = \mathrm{Tr}_{q,2}(a)^t + \mathrm{Tr}_{q,2}(a) = 0. \tag{4.3.5}$$

Now, define a *half-trace* function $H : \mathbb{F}_q \to \mathbb{F}_q$ as follows[1]

$$H(u) = \sum_{i=0}^{(m-1)/2} u^{2^i}. \tag{4.3.6}$$

Then $H(u)^t + H(u) = u + \mathrm{Tr}_{q,2}(u)$, and so $H(u) \in \mathbb{F}_q$ is a solution to $P_1(x) = 0$ when $\mathrm{Tr}_{q,2}(u) = 0$.

Next we prove that if $\mathrm{Tr}_{q,2}(u) = 0$ then $P_1(x) = 0$ has exactly two solutions, namely $H(u)$ and $H(u) + 1$. We first consider the case $m = 4i + 3$. Note that $q = 2^m$ and $t = \sqrt{2q} = 2^{2i+2}$. Let us fix a normal basis to represent $\mathbb{F}_q$ as an m-dimensional vector space over $\mathbb{F}_2$. In this representation, we may set

$$\begin{aligned} x &= (x_0, x_1, \ldots, x_{2i}, x_{2i+1}, x_{2i+2}, x_{2i+3}, \ldots, x_{4i+1}, x_{4i+2}), \\ x^t &= (x_{2i+1}, x_{2i+2}, \ldots, x_{4i+1}, x_{4i+2}, x_0, x_1, \ldots, x_{2i-1}, x_{2i}), \\ u &= (u_0, u_1, \ldots, u_{2i}, u_{2i+1}, u_{2i+2}, u_{2i+3}, \ldots, u_{4i+1}, u_{4i+2}). \end{aligned}$$

[1]The definition is similar to the one in [40, Section 3.6.2]

Then $P_1(x) = 0$ has a solution if and only if the linear system of equations determined by

$$\begin{aligned} x_j + x_{2i+2+j} &= u_{2i+2+j},\ 0 \le j \le 2i, && (4.3.7)\\ x_{2i+1+j} + x_j &= u_j,\ 1 \le j \le 2i+1, && (4.3.8)\\ x_{2i+2} + x_0 &= u_0 && (4.3.9) \end{aligned}$$

has a solution $X = (x_0, x_1, \ldots, x_{4i+2}) \in \mathbb{F}_2^m$. We can see from (4.3.7) and (4.3.8) that a choice of $x_0 \in \{0,1\}$ fixes x_j for all $1 \le j \le 4i+2$, and hence fixes two vectors X_0 and X_1 in $\mathbb{F}_2^m$. Now, it follows from (4.3.9) that $P_1(x) = 0$ has a solution if and only if $x_0 + x_{2i+1} = u_0$. Therefore, $P_1(x) = 0$ has at most two solutions in $\mathbb{F}_q$. In particular, when $P_1(x) = 0$ has a solution a_0 then there are exactly two solutions and the other solution is $a_1 = a_0 + 1$ since

$$P_1(a_0+1) = (a_0+1)^t + (a_0+1) + u = a_0^t + a_0 + u = 0.$$

The case $m = 4i+1$ can be similarly proven. □

Now, we are ready to describe our compression/decompression maps that achieve factor-$(4\log q/(1+\log q))$ compression.

Theorem 4.3.4. *For some fixed representation of $\mathbb{F}_q$ as an m-dimensional vector space over $\mathbb{F}_2$, let j be a coordinate position such that the vector representations of β and $\beta+1$ differ in the jth coordinate position for all $\beta \in \mathbb{F}_q$. Let $G \in \{G_\ell, G_{\bar{\ell}}\}$. Define a compression map*

$$\begin{aligned} \mathcal{C} : G \setminus \{1\} &\to \{0,1\} \times \mathbb{F}_q && (4.3.10)\\ g &\mapsto (i, b), \end{aligned}$$

where $g = g_0 + g_1\sigma$, $(g_0+1)/g_1 = a + bw$, and i is the jth bit in the vector representation of a. And define a decompression map

$$\begin{aligned} \mathcal{D} : \{0,1\} \times \mathbb{F}_q &\to G \setminus \{1\} && (4.3.11)\\ (i, b) &\mapsto (\alpha+\sigma)/(\alpha+1+\sigma), \end{aligned}$$

where $\alpha = a + bw$, and a is one of the two roots of $P_1(x, b)$ (see Theorem 4.3.2) whose jth bit when represented as a vector over $\mathbb{F}_2$ is equal to i. Then $\mathcal{C}$ and $\mathcal{D}$ are inverses of each other when they are defined. Moreover, if $\mathcal{D}(0,b) \in G$ then $\mathcal{D}(1,b) \in G$ and $\mathcal{D}(0,b)\mathcal{D}(1,b) = 1$.

Proof. It follows from Theorem 4.3.2 and Lemma 4.3.3 that $\mathcal{C}$ and $\mathcal{D}$ are inverses of each other when they are defined. Now, by Lemma 4.3.3, $P_1(x, b)$ has exactly 2 solutions a_0 and a_1 in $\mathbb{F}_q$, and $a_1 = a_0 + 1$. Note that since $g = \frac{\alpha+\sigma}{\alpha+1+\sigma} \in G$ with $\alpha = a + bw$ corresponding to (a_0, b), the element $h = \frac{\alpha+1+\sigma}{\alpha+\sigma}$ corresponds to (a_1, b) and is in fact the multiplicative inverse of g. It follows that $\mathcal{D}(0,b)\mathcal{D}(1,b) = 1$. □

Remark 4.3.5. Let $G \in \{G_\ell, G_{\bar{\ell}}\}$. We should note that $\mathcal{D}$ is only defined on a subset of $\{0,1\} \times \mathbb{F}_q$. In particular, $\mathcal{D}(i,b) \in G \setminus \{1\}$ if $(i,b) = \mathcal{C}(g)$ for some $g \in G \setminus \{1\}$. It would be interesting to determine whether $P_1(x,b) = 0$ has any solution in $\mathbb{F}_q$ when $b \in \mathbb{F}_q$ and $(i,b) \neq \mathcal{C}(g)$ for all $i \in \{0,1\}$ and $g \in G \setminus \{1\}$.

Remark 4.3.6. The polynomials $P_0(x,y)$ and $P_1(x,y)$ are both of degree $(t+1) \approx \sqrt{q}$ which is in accordance with Remark 4.2.1. However, $P_1(x,b)$ is very sparse and moreover it is easy to find a root $a \in \mathbb{F}_q$, in contrast to what one would expect in general for high-degree polynomials.

4.4 Factor-6 compression in characteristic three

Let $q = 3^m$, $m \equiv 5 \pmod{12}$, $t = \sqrt{3q}$ and $\ell = q + 1 - t$. Then

$$\begin{aligned} q^6 - 1 &= (q^3 - 1)(q^3 + 1) \\ &= (q^3 - 1)(q + 1)(q^2 - q + 1) \\ &= (q^3 - 1)(q + 1)(q + 1 - t)(q + 1 + t). \end{aligned}$$

Let $G_\ell \subset G_{q,6} \subset \mathbb{F}_{q^6}^*$ be subgroups such that $|G_{q,6}| = q^2 - q + 1$ and $|G_\ell| = \ell$. Since $f(w) = w^3 - w - 1$ has splitting field $\mathbb{F}_{3^3}$ and $\gcd(3, m) = 1$, f is irreducible over $\mathbb{F}_q$ and we set $\mathbb{F}_{q^3} = \mathbb{F}_q[w]/(w^3 - w - 1)$. We also let $c_0 \in \mathbb{F}_{q^3}$ be a quadratic non-residue and set $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[\sigma]/(\sigma^2 - c_0)$.

Lemma 4.4.1. *Let $\mathbb{F}_{q^3} = \mathbb{F}_q[w]/(w^3 - w - 1)$ and $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[\sigma]/(\sigma^2 - c_0)$ where $c_0 \in \mathbb{F}_{q^3}$ be a quadratic non-residue. Then*

$$\begin{aligned} \sigma^t &= c_1\sigma, \\ \sigma^q &= c_2\sigma, \\ \sigma^{q^2} &= c_3\sigma, \\ w^t &= w, \\ w^{2t} &= w^2, \\ w^q &= w + 2, \\ w^{2q} &= w^2 + w + 1, \\ w^{q^2} &= w + 1, \end{aligned}$$

for some $c_1, c_2, c_3 \in \mathbb{F}_{q^3}$.

Proof. The equalities can be proven by using the defining equations of σ and w, and noting that $w^{3^{2k}} = w + 2k$ and $w^{3^{2k+1}} = w + 1 + 2k$. □

We furthermore assume throughout this section that $c_i = u_{3i} + u_{3i+1}w + u_{3i+2}w^2$ for $i = 0, 1, 2, 3$, where $u_j \in \mathbb{F}_q$.

Let $g = g_0 + g_1\sigma \in G_{q,6}$. We already know from Section 4.1 that if $g \neq \pm 1$ then g can be compressed to an element $\alpha = (g_0 + 1)/g_1 \in \mathbb{F}_{q^3}$, and that a compressed $\alpha \in \mathbb{F}_{q^3}$ can be decompressed to obtain $g = (\alpha + \sigma)/(\alpha - \sigma) \in G_{q,6} \setminus \{\pm 1\}$. Our objective is to show that $g \in G_\ell$ can be compressed to $c \in \mathbb{F}_q$, and that a compressed $c \in \mathbb{F}_q$ can be decompressed to obtain $g \in G_\ell \setminus \{\pm 1\}$. The following theorem plays a key role.

Theorem 4.4.2. *Let $g = (\alpha + \sigma)/(\alpha - \sigma) \in G_\ell \setminus \{\pm 1\}$ where $\alpha = a + bw + cw^2$ for some $a, b, c \in \mathbb{F}_q$. Then $(x_1, x_2, x_3, x_4, x_5, x_6) = (a, a^t, b, b^t, c, c^t)$ is a root of each f_i and g_i for $i = 0, 1, 2$, where*

$$\begin{aligned} f_0 \;=\;& u_0u_3u_6 + u_0u_4u_8 + u_0u_5u_7 + u_1u_3u_8 + u_1u_4u_7 + u_1u_5u_6 \\ &+u_1u_5u_8 + u_2u_3u_7 + u_2u_4u_6 + u_2u_4u_8 + u_2u_5u_7 + u_2u_5u_8 \\ &+u_3x_1^2 + (u_4 + 2u_5)x_3^2 + (u_3 + 2u_4 + 2u_5)x_5^2 + (2u_6 + 2)x_1x_2 \\ &+(2u_7 + 2u_8 + 2)x_5x_6 + (2u_6 + 2u_8 + 2)x_3x_6 + (2u_8 + 1)x_2x_3 \\ &+(2u_7 + 2)x_2x_5 + (2u_6 + 2u_8 + 2)x_4x_5 + (2u_3 + 2u_5)x_1x_3 \\ &+(u_3 + 2u_4 + u_5)x_1x_5 + 2u_3x_3x_5 + 2u_7x_3x_4 + 2u_7x_1x_6 \\ &+2u_8x_1x_4, \end{aligned}$$

$$\begin{aligned} f_1 \;=\;& u_0u_3u_7 + u_0u_4u_6 + u_0u_4u_8 + u_0u_5u_7 + u_0u_5u_8 + u_1u_3u_6 \\ &+u_1u_3u_8 + u_1u_4u_7 + u_1u_4u_8 + u_1u_5u_6 + u_1u_5u_7 + u_1u_5u_8 \\ &+u_2u_3u_7 + u_2u_3u_8 + u_2u_4u_6 + u_2u_4u_7 + u_2u_4u_8 + u_2u_5u_6 \\ &+u_2u_5u_7 + 2u_2u_5u_8 + u_4x_1^2 + (2u_3 + u_4)x_3^2 + (2u_3 + u_5)x_5^2 \\ &+(2u_6 + 2u_7 + u_8 + 1)x_5x_6 + (2u_7 + 2u_8 + 1)x_3x_4 \\ &+(2u_6 + 2u_7 + 2u_8 + 2)x_3x_6 + (2u_6 + 2u_8 + 2)x_2x_3 \\ &+(2u_7 + 2u_8 + 2)x_2x_5 + (2u_6 + 2u_7 + 2u_8 + 1)x_4x_5 \\ &+(2u_7 + 2u_8)x_1x_6 + (2u_6 + 2u_8 + 2)x_1x_4 + u_3x_1x_5 \\ &+(2u_3 + 2u_4 + 2u_5)x_1x_3 + 2u_4x_3x_5 + 2u_7x_1x_2, \end{aligned}$$

$$
\begin{aligned}
f_2 \;=\;& u_0u_3u_8 + u_0u_4u_7 + u_0u_5u_6 + u_0u_5u_8 + u_1u_3u_7 + u_1u_4u_6 \\
&+u_1u_4u_8 + u_1u_5u_7 + u_1u_5u_8 + u_2u_3u_6 + u_2u_3u_8 + u_2u_4u_7 \\
&+u_2u_4u_8 + u_2u_5u_6 + u_2u_5u_7 + u_2u_5u_8 + 2u_7x_2x_3 + u_5x_1^2 \\
&+(u_3 + 2u_4 + u_5)x_3^2 + (2u_3 + 2u_4)x_5^2 + (2u_4 + 2u_5)x_1x_3 \\
&+(2u_3 + u_4)x_1x_5 + 2u_8x_1x_2 + 2u_5x_3x_5 + 2u_7x_1x_4 \\
&+(2u_6 + 2u_7 + 2u_8 + 1)x_5x_6 + (2u_6 + 2u_8 + 2)x_3x_4 \\
&+(2u_7 + 2u_8 + 1)x_3x_6 + (2u_6 + 2u_8 + 2)x_2x_5 \\
&+(2u_7 + 2u_8 + 2)x_4x_5 + (2u_6 + 2u_8 + 2)x_1x_6,
\end{aligned}
$$

$$
\begin{aligned}
g_0 \;=\;& u_2u_7u_{11} + u_2u_8u_{10} + u_2u_8u_{11} + u_0u_6u_9 + u_1u_7u_{10} + u_0u_7u_{11} \\
&+u_0u_8u_{10} + u_1u_6u_{11} + u_2u_7u_9 + u_1u_8u_9 + u_1u_8u_{11} + u_2u_6u_{10} \\
&+(2u_9 + 2 + u_6)x_1^2 + (u_9 + u_6 + 2u_8 + u_{11})x_3x_1 \\
&+(2u_9 + 1 + 2u_8 + u_{10} + 2u_{11} + 2u_7 + u_6)x_1x_5 \\
&+(u_7 + u_8 + 2u_{10} + u_{11} + 1)x_3^2 \\
&+(2u_6 + 2u_7 + 2u_9 + u_{10} + u_{11} + 2)x_5^2 + (2u_6 + u_9 + 1)x_3x_5
\end{aligned}
$$

$$
\begin{aligned}
g_1 \;=\;& u_0u_6u_{10} + u_0u_7u_9 + u_0u_7u_{11} + u_0u_8u_{10} + u_0u_8u_{11} + u_1u_6u_9 \\
&+u_1u_6u_{11} + u_1u_7u_{10} + u_1u_7u_{11} + u_1u_8u_9 + u_1u_8u_{10} + u_1u_8u_{11} \\
&+u_2u_6u_{10} + u_2u_6u_{11} + u_2u_7u_9 + u_2u_7u_{10} + u_2u_7u_{11} + u_2u_8u_9 \\
&+u_2u_8u_{10} + 2u_2u_8u_{11} + (u_7 + 2u_{10})x_1^2 \\
&+(u_6 + u_7 + 2u_8 + u_9 + 2u_{10})x_3^2 \\
&+(u_7 + 2u_8 + u_9 + 2u_{11} + 2)x_5^2 \\
&+(2u_6 + u_7 + 2u_8 + u_9 + u_{10} + u_{11} + 1)x_1x_3 \\
&+(2u_6 + u_8 + 2u_9)x_1x_5 + (2u_7 + u_{10})x_3x_5,
\end{aligned}
$$

$$
\begin{aligned}
g_2 \;=\;& u_1u_6u_{10} + u_1u_7u_9 + u_1u_7u_{11} + u_1u_8u_{10} + u_1u_8u_{11} + u_2u_6u_9 \\
&+u_2u_6u_{11} + u_2u_7u_{10} + u_2u_7u_{11} + u_0u_6u_{11} + u_0u_7u_{10} + u_0u_8u_9 \\
&+u_0u_8u_{11} + u_2u_8u_9 + u_2u_8u_{10} + u_2u_8u_{11} + (u_8 + 2u_{11})x_1^2 \\
&+(2u_7 + u_8 + u_{10} + u_{11})x_1x_3 \\
&+(2u_6 + 2u_7 + u_9 + 2u_{10} + 1)x_1x_5 \\
&+(u_6 + u_7 + u_8 + 2u_9 + u_{10} + 2u_{11} + 2)x_3^2 \\
&+(2u_6 + u_8 + u_9 + u_{10} + 1)x_5^2 + (2u_8 + u_{11})x_3x_5,
\end{aligned}
$$

where $c_i = u_{3i} + u_{3i+1}w + u_{3i+2}w^2$*, for* $i = 0, 1, 2, 3$*, are as specified in Lemma 4.4.1.*

Proof. Let $g = (\alpha + \sigma)/(\alpha - \sigma) \in G_\ell \setminus \{\pm 1\}$. Expanding the equations $g^{q^2-q+1} = 1$ and $g^{q+1-t} = 1$ and simplifying using Lemma 4.4.1 yields the polynomials $f_i, g_i \in \mathbb{F}_q[x_1, x_2, \ldots, x_6]$ for $i = 0, 1, 2$ such that $(x_1, x_2, x_3, x_4, x_5, x_6) = (a, a^t, b, b^t, c, c^t)$ is a root of each f_i and g_i, as required. □

Theorem 4.4.2 suggests that one can compress an element $g \in G_\ell \setminus \{\pm 1\}$ to an element $c \in \mathbb{F}_q$. Given a compressed representation c of an element g, one might reconstruct g by finding a common root (a, a^t, b, b^t, c, c^t) of the f_i and g_i. This may be achieved by constructing a Groebner basis of the ideal in $\mathbb{F}_q[x_1, x_2, \ldots, x_6]$ generated by f_i and g_i evaluated at $x_5 = c$, $x_6 = c^t$ for $i = 0, 1, 2$. The next corollary shows that this is indeed possible in the case that $c_0 = -1$.

Corollary 4.4.3. *Let* $\mathbb{F}_{q^3} = \mathbb{F}_q[w]/(w^3 - w - 1)$ *and* $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[\sigma]/(\sigma^2 + 1)$*. Let* f_i, g_i *be as in Theorem 4.4.2. Then a Groebner basis of the ideal* $\langle f_0, f_1, f_2, g_0, g_1, g_2 \rangle$ *in* $\mathbb{F}_q[x_1, x_2, \ldots, x_6]$ *is*

$$\begin{aligned}
P_1 &= x_1 + 2x_3^2x_4 + 2x_3x_4x_5 + x_3 + x_5^2x_6 + 2x_5, \\
P_2 &= x_2 + 2x_3^3x_4^2 + 2x_4^2x_5^3 + x_4x_5^3x_6 + x_5^3x_6^2 + 2x_6, \\
P_3 &= x_3^2x_4x_5 + 2x_3^2 + x_3x_4x_5^2 + 2x_3x_5 + 2x_5^3x_6 + 2x_5^2 + 2, \\
P_4 &= x_3x_4^2x_5^2 + x_3x_4x_5 + x_3 + 2x_4^2x_5^3 + 2x_4x_5^3x_6 + 2x_4x_5^2 \\
&\quad + 2x_5^3x_6^2 + 2x_5 + 2x_6, \\
P_5 &= x_3x_6 + 2x_4x_5 + 2x_5x_6 + 1, \\
P_6 &= x_4^3x_5^3 + 2x_4x_5^3x_6^2 + 2x_5^3x_6^3 + 2x_6^2 + 2.
\end{aligned}$$

Proof. If one sets $c_0 = -1$ in Theorem 4.4.2 then $g_1 = g_2 = 0$, and the polynomials f_i and g_0 simplify to

$$\begin{aligned}
f_0 &= 2x_1^2 + x_1x_3 + 2x_1x_5 + x_2x_3 + 2x_2x_5 + x_3x_5 + 2x_5^2 + 2x_5x_6 + 2, \\
f_1 &= x_1x_3 + 2x_1x_5 + 2x_2x_5 + x_3^2 + x_3x_4 + 2x_4x_5 + x_5^2 + 2x_5x_6, \\
f_2 &= x_1x_5 + 2x_3^2 + x_3x_6 + 2x_4x_5 + x_5^2 + 2x_5x_6, \\
g_0 &= 2x_1x_5 + x_3^2 + 2x_5^2 + 1.
\end{aligned}$$

It can be verified using Magma with the commands

```
R < x1, x2, x3, x4, x5, x6 >:= PolynomialRing(FiniteField(3), 6);
B := [R!f_0, R!f_1, R!f_2, R!g_0];
I := ideal < R|B >;
GroebnerBasis(I);
```

that a Groebner basis of the ideal $\langle f_0, f_1, f_2, g_0\rangle$ in $\mathbb{F}_q[x_1, x_2, \ldots, x_6]$ is determined by the P_i's, as required. □

4.4.1 Decompression procedure

Let $\mathbb{F}_{q^3} = \mathbb{F}_q[w]/(w^3-w-1)$ and $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[\sigma]/(\sigma^2+1)$. Let $g = (\alpha+\sigma)/(\alpha-\sigma) \in G_\ell \setminus \{\pm 1\}$ where $\alpha = a + bw + cw^2$ for some $a, b, c \in \mathbb{F}_q$. By Theorem 4.4.2 and Corollary 4.4.3, b^t must be a root of

$$P_6(x_4) = c^3 x_4^3 + 2c^{2t+3} x_4 + 2(c^{3(t+1)} + c^{2t} + 1). \tag{4.4.1}$$

In fact, there are exactly three roots of $P_6(x_4)$ in $\mathbb{F}_q$, and if r is a root then the other two roots are given by $r \pm c^t$. Therefore, if c is given, b^t can be determined uniquely up to 3 elements, that is, $b^t \in \{r, r - c^t, r + c^t\}$. Once b^t is fixed, one can solve for b uniquely by using $P_5(x_3) = 0$, where $P_5(x_3)$ is obtained by evaluating P_5 at $x_4 = b^t$, $x_5 = c$, $x_6 = c^t$ (see Corollary 4.4.3), or by using the fact that $b \mapsto b^t$ is a Frobenius map. Having determined b, b^t, c and c^t we can use $P_2(x_2) = 0$, where $P_2(x_2)$ is obtained by evaluating P_2 at $x_3 = b$, $x_4 = b^t$, $x_5 = c$, $x_6 = c^t$ (see Corollary 4.4.3), to solve for a^t uniquely. Finally, a can be determined either by using $P_1(x_1) = 0$, where $P_1(x_1)$ is obtained by evaluating P_1 at $x_3 = b$, $x_4 = b^t$, $x_5 = c$, $x_6 = c^t$ (see Corollary 4.4.3), or by using the fact that $a \mapsto a^t$ is a Frobenius map.

To summarize, suppose that $g \in G_\ell \setminus \{\pm 1\}$ and $g = (\alpha+\sigma)/(\alpha-\sigma)$ with $\alpha = a+bw+cw^2$. If c is given, then the three pairs (x_{1h}, x_{3h}), $h = 1, 2, 3$ can be efficiently determined such that $(a, b, c) \in \{(x_{1h}, x_{3h}, c) : \ h = 1, 2, 3\}$. In fact, one can check that $c \neq 0$ and

$$\{(x_{1h}, x_{3h}, c) : \ h = 1, 2, 3\} = \{(a, b, c), (a - b + c, b + c, c), (a + b + c, b - c, c)\}.$$

Suppose now that we have fixed some representation of $\mathbb{F}_q$ as an m-dimensional vector space over $\mathbb{F}_3$. Then there must exist a *smallest index* j such that exactly one of x_{3h}'s j'th trit is equal to b's j'th trit, say $i \in \{0, 1, 2\}$, when they are represented as vectors over $\mathbb{F}_3$. This yields one-to-one compression/decompression maps that achieve factor-$(6 \log q)/(2 + \log q)$ compression.

Theorem 4.4.4. *Define a compression map*

$$\begin{aligned} \mathcal{C} : G_\ell \setminus \{\pm 1\} &\to \{0, 1, 2\} \times \mathbb{F}_q \\ g &\mapsto (i, c), \end{aligned} \tag{4.4.2}$$

where $g = g_0+g_1\sigma$, $(g_0+1)/g_1 = a+bw+cw^2$, and i is defined above. Define a decompression map

$$\begin{aligned} \mathcal{D} : \{0, 1, 2\} \times \mathbb{F}_q &\to G_\ell \setminus \{\pm 1\} \\ (i, c) &\mapsto (\alpha + \sigma)/(\alpha - \sigma), \end{aligned} \tag{4.4.3}$$

where $\alpha = a + bw + cw^2$, and a, b can be constructed as described above. Then $\mathcal{C}$ and $\mathcal{D}$ are inverses of each other when they are defined. Moreover, if $\mathcal{D}(0, c) \in G_\ell \setminus \{\pm 1\}$ then $\mathcal{D}(i, b) \in G_\ell \setminus \{\pm 1\}$ for $i = 1, 2$, and $\mathcal{D}(0, c)\mathcal{D}(1, c)\mathcal{D}(2, c) = 1$.

Proof. It is clear from our arguments above that $\mathcal{C}$ and $\mathcal{D}$ are inverses of each other when they are defined. Now, let $c \in \mathbb{F}_q^*$ be such that $g = (\alpha + \sigma)/(\alpha - \sigma) \in G_\ell \setminus \{\pm 1\}$, where $\alpha = a + bw + cw^2$. Let i_1 be the jth trit of b, where j is the smallest index such that if $(b + c)$'s j'th trit is i_2 and $(b - c)$'s j'th trit is i_3, then i_1, i_2, i_3 are pairwise different. It follows from our arguments above that the decompression function satisfies $\mathcal{D}(i_h, c) = g_h$, where $g_h = (\alpha_h - \sigma)/(\alpha_h + \sigma)$, and $\alpha_1 = a + bw + cw^2, \alpha_2 = (a - b + c) + (b + c)w + cw^2, \alpha_3 = (a + b + c) + (b - c)w + cw^2$. Moreover, one can check that $g_1 = g, g_2 = g^{-q} = g^{q^4}$, and $g_3 = g^{q^2}$, that is $g_1 g_2 g_3 = 1$, as required. □

Remark 4.4.5. We should note that $\mathcal{D}$ is only defined on a subset of $\{0, 1, 2\} \times \mathbb{F}_q$. In particular, $\mathcal{D}(i, c) \in G_\ell \setminus \{\pm 1\}$ if $(i, c) = \mathcal{C}(g)$ for some $g \in G_\ell \setminus \{\pm 1\}$. It would be interesting to determine whether $P_6(x) = 0$ has any solution in $\mathbb{F}_q$ when $c \in \mathbb{F}_q$ and $(i, c) \neq \mathcal{C}(g)$ for all $i \in \{0, 1, 2\}$ and $g \in G_\ell \setminus \{\pm 1\}$.

Remark 4.4.6. It would be interesting to prove similar results for $q = 3^m$ where $m \not\equiv 5 \pmod{12}$, and for any quadratic non-residue $c_0 \in \mathbb{F}_{q^3}$, $c_0 \neq -1$. The main difficulty when $c_0 \neq -1$ seems to be that the polynomials f_i, g_i are defined strictly over $\mathbb{F}_q$ rather than over $\mathbb{F}_3$ which is the case when $c_0 = -1$.

4.5 Factor-4 compression and exponentiation algorithms

In this section, we analyze the efficiency of the compression and decompression methods proposed in Section 4.3. The efficiency of these methods matters because given a compressed representation of an element, one can consider a variety of exponentiation algorithms that can work directly with that compressed representation, or with partially or fully decompressed representations of the element.

We first show that compression and decompression can be achieved at a negligible cost. Then we describe two exponentiation algorithms and provide a performance comparison.

4.5.1 Compression/decompression costs

Let $q = 2^m$, m odd, $t = \sqrt{2q}$ and $\ell = q + 1 - t$. Let $G_\ell \subset \mathbb{F}_{q^4}^*$ be the subgroup with $|G_\ell| = \ell$. Let $\mathbb{F}_{q^2} = \mathbb{F}_q[w]/(w^2 + w + c_0)$ and $\mathbb{F}_{q^4} = \mathbb{F}_{q^2}[\sigma]/(\sigma^2 + \sigma + c_1)$, where $\mathrm{Tr}_{q,2}(c_0) = \mathrm{Tr}_{q^2,2}(c_1) = 1$. We further assume that $\mathbb{F}_q$ is represented as an m-dimensional vector space over $\mathbb{F}_2$ via a polynomial basis $\{1, z, \ldots, z^{m-1}\}$.

We first show that the compression and decompression maps described in Theorem 4.3.4 are very efficiently computable.

Lemma 4.5.1. *Let $P_1(x) = x^t + x + u \in \mathbb{F}_q[x]$. If $P_1(x) = 0$ has a solution in $\mathbb{F}_q$ then it can be computed at a cost of $(m-1)/2$ squarings and $(m-1)/2$ additions in $\mathbb{F}_q$. If storage for m $\mathbb{F}_q$-elements is available then finding the $\mathbb{F}_q$-solutions of $P_1(x) = 0$ in $\mathbb{F}_q$ requires on average $m/2$ additions in $\mathbb{F}_q$.*

Proof. Let $u = \sum_{i=0}^{m-1} u_i z^i$ and suppose that $P_1(x) = x^t + x + u = 0$ has a solution in $\mathbb{F}_q$. It follows from Lemma 4.3.3 that the solutions in $\mathbb{F}_q$ are given by $H(u)$ and $H(u) + 1$, where $H(u) = \sum_{i=0}^{(m-1)/2} u^{2^i}$, which can be computed at a cost of $(m-1)/2$ squarings and $(m-1)/2$ additions in $\mathbb{F}_q$. If one can store $H(z^i)$ for $0 \leq i < m$ then

$$H(u) = \sum_{i=0}^{m-1} u_i H(z^i)$$

can be computed at a cost of $m/2$ additions on average. □

Theorem 4.5.2. *Let $\mathcal{C}$ and $\mathcal{D}$ be compression and decompression maps, respectively, as described in Theorem 4.3.4. Then compression via $\mathcal{C}$ requires 1 division in $\mathbb{F}_{q^2}$ and decompression via $\mathcal{D}$ requires $(m-1)/2$ squarings and $(m-1)/2$ additions in $\mathbb{F}_q$, and 1 division in $\mathbb{F}_{q^4}$. If storage for m $\mathbb{F}_q$-elements is available then decompression requires on average $m/2$ additions in $\mathbb{F}_q$ and 1 division in $\mathbb{F}_{q^4}$.*

Proof. The proof follows from Theorem 4.3.4 and Lemma 4.5.1. □

4.5.2 Exponentiation algorithms

Recall that in our compression method, given $g = g_0 + g_1\sigma \in G_\ell \setminus \{1\}$, we first compress g to $\alpha = (g_0 + 1)/g_1 = a + bw$, and then compress α to (i, b) where $i \in \{0, 1\}$. By Theorem 4.5.2, compressing g to (i, b) and decompressing (i, b) to α (and to g) can be achieved at a negligible cost. In this context, we call α a *half-compressed* element.

We present two exponentiation algorithms to compute g^e given $\mathcal{C}(g) = (i, b)$ and $e \in \mathbb{Z}$. The first exponentiation algorithm, which we call *HCTBE* (Half-Compressed Torus-Based Exponentiation), partially decompresses (i, b) to α and then uses a multiplication formula for half-compressed elements. The output is then compressed to obtain $\mathcal{C}(g^e)$. The second algorithm, which we call *FDDE* (Fully-Decompressed Direct Exponentiation Algorithm), fully decompresses (i, b) to g and uses a conventional square-and-multiply exponentiation algorithm in $\mathbb{F}_{q^4}$.

The HCTBE algorithm

The algorithm makes use of the multiplication formula (4.1.3) to compute $\mathcal{C}(g^e)$. The formula requires an inversion in $\mathbb{F}_{q^2}$ that makes the exponentiation algorithm quite costly if one tries to use (4.1.3) directly. However, the problem can be overcome as follows. If $g = g_0 + g_1\sigma$, $h = h_0 + h_1\sigma \in G_\ell \setminus \{1\}$ are represented by $\alpha = (g_0+1)/g_1$, $\beta = (h_0+1)/h_1 \in \mathbb{F}_{q^2}$, respectively, then we have

$$\begin{aligned} g \cdot h &= \left(\frac{\alpha+\sigma}{\alpha+1+\sigma}\right)\left(\frac{\beta+\sigma}{\beta+1+\sigma}\right) \\ &= \frac{\alpha\beta + c_1 + (\alpha+\beta+1)\sigma}{\alpha\beta + c_1 + \alpha + \beta + 1 + (\alpha+\beta+1)\sigma}. \end{aligned}$$

In other words, if the product of any two elements in G_ℓ is computed by this formula then the result will be of the form

$$\frac{x+y\sigma}{x+y+y\sigma}, \text{ for some } x, y \in \mathbb{F}_{q^2}. \tag{4.5.1}$$

In particular, given $\mathcal{C}(g) = (i, b)$ and $e \in \mathbb{Z}$, one can first decompress (i, b) to α, and then perform an exponentiation to compute $\mathcal{C}(g^e)$ by using the formulas

$$\begin{aligned} \left(\frac{x+y\sigma}{x+y+y\sigma}\right)^2 &= \frac{x^2 + y^2c_1 + y^2\sigma}{x^2 + y^2c_1 + y^2 + y^2\sigma}, \\ \left(\frac{\alpha+\sigma}{\alpha+1+\sigma}\right)\left(\frac{x+y\sigma}{x+y+y\sigma}\right) &= \frac{\alpha x + yc_1 + (\alpha y + x + y)\sigma}{\alpha x + yc_1 + \alpha y + x + y + (\alpha y + x + y)\sigma}, \end{aligned}$$

in the *square* and *multiply* steps of the exponentiation algorithm. Note that by (4.5.1) it suffices to only keep track of the numerator during the computations, and to do a single division in $\mathbb{F}_{q^4}$ to obtain g^e and finally its compressed value $\mathcal{C}(g^e)$. Our discussion yields Algorithm 4.1.

Assuming that $c_1 \in \mathbb{F}_{q^2}$ is chosen so that the cost of multiplying an element by c_1 is negligible, the cost of the squaring step (step 5), and the cost of the multiplication step (step 7) in Algorithm 4.1 is approximately 2 squarings in $\mathbb{F}_{q^2}$ and 2 multiplications in $\mathbb{F}_{q^2}$, respectively.

The FDDE algorithm

After decompressing $\mathcal{C}(g) = (i, b)$ to $g = g_0 + g_1\sigma$, we use a conventional square-and-multiply exponentiation algorithm as described in Algorithm 4.2. Since

$$\begin{aligned} (x+y\sigma)^2 &= x^2 + y^2c + y^2\sigma, \\ (g_0 + g_1\sigma)(x+y\sigma) &= g_0x + g_1yc + (g_0y + g_1x + g_1yc)\sigma, \end{aligned}$$

Algorithm 4.1 The HCTBE exponentiation algorithm
Input: $\mathcal{C}(g)$ and e
Output: $\mathcal{C}(g^e)$

1: Write $e = \sum_{i=0}^{s-1} b_i 2^i$ where $b_i \in \{0, 1\}$ and $b_{s-1} = 1$
2: Decompress $\mathcal{C}(g)$ to α by using Theorem 4.3.4
3: $x \leftarrow \alpha$, $y \leftarrow 1$
4: **for** i from $s-2$ down to 0 **do**
5: $\quad y' \leftarrow y^2,\ x' \leftarrow x^2 + y'c_1$
6: $\quad$ **if** $b_i = 1$ **then**
7: $\quad\quad x' \leftarrow \alpha x + yc_1,\ y' \leftarrow \alpha y + x + y$
8: $\quad$ **end if**
9: $\quad x \leftarrow x',\ y \leftarrow y'$
10: **end for**
11: $g' \leftarrow (x + y\sigma)/(x + y + y\sigma)$
12: Compress (g') to $\mathcal{C}(g') = (i', b')$, by using Theorem 4.3.4
13: Output (i', b')

each squaring step (step 5) in Algorithm 4.2 requires 2 squarings in $\mathbb{F}_{q^2}$. Using Karatsuba's technique, each multiplication step (steps 7-8) requires 3 multiplications in $\mathbb{F}_{q^2}$. We assume that $c_1 \in \mathbb{F}_{q^2}$ is chosen appropriately so that the cost of multiplying an element by c_1 is negligible.

A comparison with trace-based exponentiation

In Chapter 2, we have shown that it is possible to compress elements of G_ℓ by a factor 4 by identifying an element $g \in G_\ell$ with its trace $\mathrm{Tr}_{q^4,q}(g)$. Given $\mathrm{Tr}_{q^4,q}(g)$ and an integer e, five exponentiation algorithms were proposed and analyzed in Chapter 2 to compute $\mathrm{Tr}_{q^4,q}(g^e)$. The algorithms are based on the following ideas:

1. Use $\mathrm{Tr}_{q^4,q}(g)$ directly and perform computations in $\mathbb{F}_q$ (Algorithm 2.1 in Chapter 2).

2. First decompress $\mathrm{Tr}_{q^4,q}(g)$ to $\mathrm{Tr}_{q^4,q^2}(g)$. Then use $\mathrm{Tr}_{q^4,q^2}(g)$ directly and perform computations in $\mathbb{F}_{q^2}$ (Algorithm 2.2 in Chapter 2).

3. First decompress $\mathrm{Tr}_{q^4,q}(g)$ to g and perform computations in $\mathbb{F}_{q^4}$ (Algorithm DDE in Chapter 2).

4. First decompress $\mathrm{Tr}_{q^4,q}(g)$ to $\mathrm{Tr}_{q^4,q^2}(g)$. Then use $\mathrm{Tr}_{q^4,q^2}(g)$ to construct a copy of $\mathbb{F}_{q^4}$ based on the minimal polynomial of g over $\mathbb{F}_{q^2}$, and perform computations in $\mathbb{F}_{q^4}$ (Algorithm BPV-I in Chapter 2).

Algorithm 4.2 The FDDE exponentiation algorithm
Input: $\mathcal{C}(g)$ and e
Output: $\mathcal{C}(g^e)$

Write $e = \sum_{i=0}^{s-1} b_i 2^i$ where $b_i \in \{0, 1\}$ and $b_{s-1} = 1$
Decompress $\mathcal{C}(g)$ to $g = g_0 + g_1\sigma$ by using Theorem 4.3.4
$x \leftarrow g_0$, $y \leftarrow g_1$
for i from $s-2$ down to 0 **do**
  $y' \leftarrow y^2$, $x' \leftarrow x^2 + y'c_1$
  if $b_i = 1$ **then**
    $u_0 \leftarrow (g_0 + g_1)(x' + y')$, $u_1 \leftarrow g_0 x'$, $u_2 \leftarrow g_1 y'$, $u_3 \leftarrow u_2 c_1$
    $x' \leftarrow u_1 + u_3$, $y' \leftarrow x' + u_0 + u_2$
  end if
  $x \leftarrow x'$, $y \leftarrow y'$
end for
$g' \leftarrow (x + y\sigma)$
Compress (g') to $\mathcal{C}(g') = (i', b')$, by using Theorem 4.3.4
Output (i', b')

5. Use $\mathrm{Tr}_{q^4,q}(g)$ to construct a copy of $\mathbb{F}_{q^4}$ based on the minimal polynomial of g over $\mathbb{F}_q$, and perform computations in $\mathbb{F}_{q^4}$ (Algorithm BPV-II in Chapter 2).

If a decompression is performed then it is the most expensive step in these algorithms. Therefore, the algorithms based on (1) and (5) are overall faster than the algorithms based on (2), (3) and (4). In particular, Algorithm 2.1 in Chapter 2 was reported to be the fastest exponentiation algorithm in the case of using a general base $\mathrm{Tr}_{q^4,q}(g)$, and its performance was further improved in Chapter 3 (see Algorithm 3.1). However, once decompression can be performed in advance, such as in the case of using a fixed base $\mathrm{Tr}_{q^4,q}(g)$, then the algorithm based on (3) is the fastest.

Note that by Theorem 4.5.2, given $\mathcal{C}(g)$ for some $g \in G_\ell \setminus \{1\}$, one can recover g (and also $\mathrm{Tr}_{q^4,q}(g)$ and $\mathrm{Tr}_{q^4,q^2}(g)$) at a negligible cost. Hence, it is more advantageous to use $\mathcal{C}(g)$ instead of $\mathrm{Tr}_{q^4,q}(g)$. For example, using $\mathcal{C}(g)$, we can obtain faster exponentiation algorithms than the trace-based exponentiation algorithms in the case of a general base $\mathrm{Tr}_{q^4,q}(g)$, by simply computing $\mathrm{Tr}_{q^4,q}(g)$ from $\mathcal{C}(g)$ and adapting an algorithm based on (3).

4.6 Factor-6 compression and exponentiation algorithms

This section is analogous to Section 4.5. We analyze the efficiency of the compression and decompression methods proposed in Section 4.4. We first show that compression and decompression can be achieved at a negligible cost, and then describe two exponentiation algorithms and provide a performance comparison.

4.6.1 Compression/decompression costs

Let $q = 3^m$, m odd, $t = \sqrt{3q}$ and $\ell = q + 1 - t$. Let $G_\ell \subset \mathbb{F}_{q^6}^*$ be the subgroup with $|G_\ell| = \ell$. Let $\mathbb{F}_{q^3} = \mathbb{F}_q[w]/(w^3 - w - 1)$. and $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[\sigma]/(\sigma^2 - c_0)$ where c_0 is a quadratic non-residue in $\mathbb{F}_{q^3}$. We further assume that $\mathbb{F}_q$ is represented as an m-dimensional vector space over $\mathbb{F}_3$ via a polynomial basis $\{1, z, \ldots, z^{m-1}\}$.

Lemma 4.6.1. *Let $P_6(x) = c^3x^3 + 2c^{2t+3}x + 2(c^{3(t+1)} + c^{2t} + 1) \in \mathbb{F}_q[x]$ for some $c \in \mathbb{F}_q$. If $P_6(x) = 0$ has a solution in $\mathbb{F}_q$ and storage of m $\mathbb{F}_q$-elements is available, then finding the $\mathbb{F}_q$-solutions requires on average $2m/3$ additions, 2 multiplications, 1 squaring and 1 division in $\mathbb{F}_q$.*

Proof. First observe that if $B \in \mathbb{F}_q$ is a quadratic non-residue and $x^3 + Bx + C = 0$ has a solution in $\mathbb{F}_q$ then all solutions are given by

$$\{r_1, r_2, r_3\} = \{(-B)^{1/2}R(D), (-B)^{1/2}(R(D) + 1), (-B)^{1/2}(R(D) + 2)\},$$

where $D = C/(-B)^{3/2}$ and $R(D)$ is a root of

$$x^3 - x + D.$$

Clearly, if $x^3 - x + D = 0$ has a solution $R(D) \in \mathbb{F}_q$ then it can be found trit-wise when a normal basis $\{\theta, \theta^3, \ldots, \theta^{3^{m-1}}\}$ is used to represent $\mathbb{F}_q$ as an m-dimensional vector space over $\mathbb{F}_3$. Let us suppose that $R(D) = \sum_{i=0}^{m-1} R_i\theta^{3^i}$ and the m $\mathbb{F}_q$-elements

$$\theta^{3^i} = \sum_{j=0}^{m-1} \theta_{ij}z^j,\ 0 \le i < m$$

are precomputed and stored. Then a solution

$$R(D) = \sum_{i=0}^{m-1} R_i \sum_{j=0}^{m-1} \theta_{ij}z^j$$

to $x^3 - x + D = 0$ is obtained in $\mathbb{F}_q$, at an average cost of $2m/3$ additions in $\mathbb{F}_q$.

Now, in order to find a solution of $P_6(x) = c^3x^3 + 2c^{2t+3}x + 2(c^{3(t+1)} + c^{2t} + 1) = 0$ in $\mathbb{F}_q$, we first compute $B = 2c^{2t+3}/c^3 = 2c^{2t}$ and $C = 2(c^{3(t+1)} + c^{2t} + 1)/c^3$. Then

$$D = C/(-B)^{3/2} = (2(c^{3(t+1)} + c^{2t} + 1)/c^{3t+3})$$

can be computed at a cost of 1 multiplication, 1 squaring and 1 division in $\mathbb{F}_q$ (we ignore the cost of addition in $\mathbb{F}_q$ and Frobenius operations). From our argument above we can find a solution of $x^3 - x + D = 0$ in $\mathbb{F}_q$ at an average cost of $2m/3$ additions in $\mathbb{F}_q$. Hence, the solutions of $P_6(x) = 0$ are given by

$$\{r_1, r_2, r_3\} = \{c^t R(D), c^t(R(D) + 1), c^t(R(D) + 2)\},$$

and can be obtained at an average cost of $2m/3$ additions, 2 multiplications, 1 squaring and 1 division in $\mathbb{F}_q$. □

From now on, we shall assume that $m \equiv 5 \pmod{12}$ and $c_0 = -1$.

Theorem 4.6.2. *Let $\mathcal{C}$ and $\mathcal{D}$ be compression and decompression maps, respectively, as described in Theorem 4.4.4. Then compression via $\mathcal{C}$ requires 1 division in $\mathbb{F}_{q^3}$ and decompression via $\mathcal{D}$ requires on average $2m/3$ additions, 2 multiplications, 1 squaring, 1 division in $\mathbb{F}_q$, and 1 division in $\mathbb{F}_{q^6}$, with a storage of m $\mathbb{F}_q$-elements.*

Proof. The proof follows from Theorem 4.4.4 and Lemma 4.6.1. □

4.6.2 Exponentiation algorithms

Recall that in our compression method, given $g = g_0 + g_1\sigma \in G_\ell \setminus \{\pm 1\}$, we first compress g to $\alpha = (g_0 + 1/g_1) = a + bw + cw^2$, by a factor 2, and then compress α to (i, c), $i \in \{0, 1, 2\}$ by a factor of 3. By Theorem 4.6.2, compressing g to (i, c), and decompressing (i, c) to α (and to g) can be achieved at a negligible cost. In this context, we call α a *half-compressed* element.

We present two exponentiation algorithms to compute g^e given $\mathcal{C}(g) = (i, c)$ and $e \in \mathbb{Z}$. The first exponentiation algorithm, which we call *HCTBE* (Half-Compressed Torus-Based Exponentiation), partially decompresses (i, c) to α and uses a multiplication formula for half-compressed elements. The output is then compressed to obtain $\mathcal{C}(g^e)$. The second algorithm, which we call *FDDE* (Fully-Decompressed Direct Exponentiation Algorithm), fully decompresses (i, c) to g and uses a conventional cube-and-multiply exponentiation algorithm in $\mathbb{F}_{q^6}$.

The HCTBE algorithm

The algorithm makes use of the multiplication formula (4.1.4) to compute $\mathcal{C}(g^e)$. If $g = g_0+g_1\sigma$, $h = h_0+h_1\sigma \in G_\ell\backslash\{\pm 1\}$ are represented by $\alpha = (g_0+1)/g_1$, $\beta = (h_0+1)/h_1 \in \mathbb{F}_{q^3}$, respectively, then we have

$$\begin{aligned} g \cdot h &= \left(\frac{\alpha+\sigma}{\alpha-\sigma}\right)\left(\frac{\beta+\sigma}{\beta-\sigma}\right) \\ &= \frac{\alpha\beta + c_0 + (\alpha+\beta)\sigma}{\alpha\beta + c_0 - (\alpha+\beta)\sigma}. \end{aligned}$$

In other words, if the product of any two elements in G_ℓ is computed by this formula then the result will be of the form

$$\frac{x+y\sigma}{x-y\sigma}, \text{ for some } x, y \in \mathbb{F}_{q^3}. \tag{4.6.1}$$

In particular, given $\mathcal{C}(g) = (i, c)$ and $e \in \mathbb{Z}$, one can first decompress (i, c) to α, and then perform an exponentiation to compute $\mathcal{C}(g^e)$ by using the formulas

$$\begin{aligned} \left(\frac{x+y\sigma}{x-y\sigma}\right)^3 &= \frac{x^3 + y^3c_0\sigma}{x^3 - y^3c_0\sigma}, \\ \left(\frac{\alpha+\sigma}{\alpha-\sigma}\right)\left(\frac{x+y\sigma}{x-y\sigma}\right) &= \frac{\alpha x + yc_0 + (\alpha y + x)\sigma}{\alpha x + yc_0 - (\alpha y + x)\sigma}, \\ \left(\frac{\alpha-\sigma}{\alpha+\sigma}\right)\left(\frac{x+y\sigma}{x-y\sigma}\right) &= \frac{\alpha x - yc_0 + (\alpha y - x)\sigma}{\alpha x - yc_0 - (\alpha y - x)\sigma} \end{aligned}$$

in the *cube* and *multiply* steps of the exponentiation algorithm. Note that by (4.6.1) it suffices to only keep track of the numerator during the computations, and to do a single division in $\mathbb{F}_{q^6}$ to obtain g^e and finally its compressed value $\mathcal{C}(g^e)$. Our discussion yields Algorithm 4.3.

Since $c_0 = -1$, the cost of the cubing step (step 5) and the cost of the multiplication step (step 7 or step 9) in Algorithm 4.3 is approximately 2 cubings in $\mathbb{F}_{q^3}$ and 2 multiplications in $\mathbb{F}_{q^3}$, respectively.

Remark 4.6.3. Granger, Page and Stam [39, Section 3.2] proposed an exponentiation algorithm that works in the quotient group $\mathbb{F}^*_{q^6}/\mathbb{F}^*_{q^3}$ where $q = 3^m$, m is odd, and mimics the mixed addition method for point multiplication on elliptic curves. Algorithm 4.1 can be seen as analogous to their algorithm. The main difference is that they identify $g = g_0 + g_1\sigma$ with $\alpha = g_0/g_1$ instead of $\alpha = (g_0+1)/g_1$ and therefore their method cannot be directly adapted to obtain a fast exponentiation algorithm in $G_\ell \subset \mathbb{F}^*_{q^6}$. In particular, it was reported in [39, Table 3] that exponentiation in $\mathbb{F}^*_{q^6}/\mathbb{F}^*_{q^3}$ is more efficient than exponentiation in G_ℓ. The HCTBE algorithm equalizes the efficiency of exponentiation algorithms in G_ℓ and $\mathbb{F}^*_{q^6}/\mathbb{F}^*_{q^3}$.

Algorithm 4.3 The HCTBE exponentiation algorithm
Input: $\mathcal{C}(g)$ and e
Output: $\mathcal{C}(g^e)$

1: Write $e = \sum_{i=0}^{s-1} b_i 3^i$ where $b_i \in \{-1, 0, 1\}$ and $b_{s-1} = 1$
2: Decompress $\mathcal{C}(g)$ to α by using Theorem 4.4.4
3: $x \leftarrow \alpha$, $y \leftarrow 1$
4: **for** i from $s-2$ down to 0 **do**
5: $\quad x' \leftarrow x^3,\ y' \leftarrow y^3 c_0$
6: $\quad$ **if** $b_i = 1$ **then**
7: $\quad\quad x' \leftarrow \alpha x + y c_0,\ y' \leftarrow (\alpha y + x)$
8: $\quad$ **else if** $b_i = -1$ **then**
9: $\quad\quad x' \leftarrow \alpha x - y c_0,\ y' \leftarrow (\alpha y - x)$
10: $\quad$ **end if**
11: $\quad x \leftarrow x',\ y \leftarrow y'$
12: **end for**
13: $g' \leftarrow (x + y\sigma)/(x - y\sigma)$
14: Compress (g') to $\mathcal{C}(g') = (i', c')$, by using Theorem 4.4.4
15: Output (i', c')

The FDDE algorithm

After decompressing $\mathcal{C}(g) = (i, b)$ to $g = g_0 + g_1\sigma$, we use a conventional cube-and-multiply exponentiation algorithm as described in Algorithm 4.4. Since

$$\begin{aligned}(x + y\sigma)^3 &= x^3 + y^3 c_0 \sigma,\\ (g_0 + g_1\sigma)(x + y\sigma) &= g_0 x + g_1 y c_0 + (g_0 y + g_1 x)\sigma,\\ (g_0 - g_1\sigma)(x + y\sigma) &= g_0 x - g_1 y c_0 + (g_0 y - g_1 x)\sigma,\end{aligned}$$

each cubing step (step 5) in Algorithm 4.4 requires 2 cubings in $\mathbb{F}_{q^3}$. Using Karatsuba's technique, each multiplication step (steps 7-8 or steps 10-11) requires 3 multiplications in $\mathbb{F}_{q^3}$ (note that $c_0 = -1$).

A comparison with trace-based exponentiation

In Chapter 2, we have shown that it is possible to compress elements of G_ℓ by a factor 6 by identifying an element $g \in G_\ell$ with its trace $\mathrm{Tr}_{q^6,q}(g)$. Given $\mathrm{Tr}_{q^6,q}(g)$ and an integer e, six exponentiation algorithms were proposed and analyzed in Chapter 2 to compute $\mathrm{Tr}_{q^6,q}(g^e)$. The performance of these six algorithms were also compared with a previously-known exponentiation algorithm XTR_3 in [78]. The algorithms are based on the following ideas:

Algorithm 4.4 The FDDE exponentiation algorithm
Input: $\mathcal{C}(g)$ and e
Output: $\mathcal{C}(g^e)$

1: Write $e = \sum_{i=0}^{s-1} b_i 3^i$ where $b_i \in \{-1, 0, 1\}$ and $b_{s-1} = 1$
2: Decompress $\mathcal{C}(g)$ to $g = g_0 + g_1\sigma$ by using Theorem 4.4.4
3: $x \leftarrow g_0$, $y \leftarrow g_1$
4: **for** i from $s-2$ down to 0 **do**
5: $\quad x' \leftarrow x^3,\ y' \leftarrow y^3 c_0$
6: $\quad$ **if** $b_i = 1$ **then**
7: $\quad\quad u_0 \leftarrow (g_0 + g_1)(x' + y'),\ u_1 \leftarrow g_0 x',\ u_2 \leftarrow g_1 y',\ u_3 \leftarrow u_2 c_0$
8: $\quad\quad x' \leftarrow u_1 + u_3,\ y' \leftarrow u_0 - (u_1 + u_2)$
9: $\quad$ **else if** $b_i = -1$ **then**
10: $\quad\quad u_0 \leftarrow (g_0 - g_1)(x' + y'),\ u_1 \leftarrow g_0 x',\ u_2 \leftarrow -g_1 y',\ u_3 \leftarrow u_2 c_0$
11: $\quad\quad x' \leftarrow u_1 + u_3,\ y' \leftarrow u_0 - (u_1 + u_2)$
12: $\quad$ **end if**
13: $\quad x \leftarrow x',\ y \leftarrow y'$
14: **end for**
15: $g' \leftarrow (x + y\sigma)$
16: Compress (g') to $\mathcal{C}(g') = (i', c')$, by using Theorem 4.4.4
17: Output (i', c')

1. Use $\mathrm{Tr}_{q^6,q}(g)$ directly and perform computations in $\mathbb{F}_q$ (Algorithm 2.3 in Chapter 2).

2. First decompress $\mathrm{Tr}_{q^6,q}(g)$ to $\mathrm{Tr}_{q^6,q^3}(g)$. Then use $\mathrm{Tr}_{q^6,q^3}(g)$ directly and perform computations in $\mathbb{F}_{q^3}$ (Algorithm 2.4 in Chapter 2).

3. First decompress $\mathrm{Tr}_{q^6,q}(g)$ to g and perform computations in $\mathbb{F}_{q^6}$ (Algorithm DDE in Chapter 2).

4. First decompress $\mathrm{Tr}_{q^6,q}(g)$ to $\mathrm{Tr}_{q^6,q^2}(g)$. Then use $\mathrm{Tr}_{q^6,q^2}(g)$ to construct a copy of $\mathbb{F}_{q^6}$ based on the minimal polynomial of g over $\mathbb{F}_{q^2}$, and perform computations in $\mathbb{F}_{q^6}$ (Algorithm BPV-I in Chapter 2).

5. First decompress $\mathrm{Tr}_{q^6,q}(g)$ to $\mathrm{Tr}_{q^6,q^3}(g)$. Then use $\mathrm{Tr}_{q^6,q^3}(g)$ to construct a copy of $\mathbb{F}_{q^6}$ based on the minimal polynomial of g over $\mathbb{F}_{q^3}$, and perform computations in $\mathbb{F}_{q^6}$ (Algorithm BPV-II in Chapter 2).

6. Use $\mathrm{Tr}_{q^6,q}(g)$ to construct a copy of $\mathbb{F}_{q^6}$ based on the minimal polynomial of g over $\mathbb{F}_q$, and perform computations in $\mathbb{F}_{q^6}$ (Algorithm BPV-III in Chapter 2).

7. First decompress $\mathrm{Tr}_{q^6,q}(g)$ to $\mathrm{Tr}_{q^6,q^2}(g)$. Then use $\mathrm{Tr}_{q^6,q^2}(g)$ directly and perform computations in $\mathbb{F}_{q^2}$ (Algorithm XTR_3 in [78]).

The algorithms based on (1), (4), (6) and (7) are overall faster than the algorithms based on (2), (3) and (5) because of the expensive decompression operations required in the latter algorithms. In particular, it was reported in Chapter 2 that XTR_3 in [78] can be further sped up and it is the fastest exponentiation algorithm for general bases $\mathrm{Tr}_{q^6,q}(g)$. However, if decompression can be precomputed, for example when the base $\mathrm{Tr}_{q^6,q}(g)$ is fixed, then the algorithm based on (3) is the fastest.

Note that by Theorem 4.6.2, given $\mathcal{C}(g)$ for some $g \in G_\ell \setminus \{\pm 1\}$, one can recover g (and also $\mathrm{Tr}_{q^6,q}(g)$, $\mathrm{Tr}_{q^6,q^2}(g)$ and $\mathrm{Tr}_{q^6,q^3}(g)$) at a negligible cost. Hence, it is more advantageous to use $\mathcal{C}(g)$ instead of $\mathrm{Tr}_{q^6,q}(g)$. For example, using $\mathcal{C}(g)$, we can obtain faster exponentiation algorithms than the trace-based exponentiation algorithms in the case of a general base $\mathrm{Tr}_{q^6,q}(g)$, by simply computing $\mathrm{Tr}_{q^6,q}(g)$ from $\mathcal{C}(g)$ and adapting an algorithm based on (3).

4.7 A comparison of exponentiation algorithms

In this section, we estimate the running times of the exponentiation algorithms discussed in Sections 4.5.2 and 4.6.2, and compare them with the fastest previously-known exponentiation algorithms. We consider the case of a general base, $\mathcal{C}(g)$ or $\mathrm{Tr}_{q^k,q}(g)$, which turns to be the most interesting case because when the base is fixed we may ignore the cost of obtaining one of $\mathcal{C}(g)$ and $\mathrm{Tr}_{q^k,q}(g)$ from the other, and hence obtain an equivalent performance in torus-based and trace-based exponentiation algorithms.

We denote by C_i, M_i, and S_i the operations of cubing, multiplication, and squaring in $\mathbb{F}_{q^i}$ for $i = 1, 2, 3$. We assume that $S_2 = 2S_1$ for characteristic two, $C_3 = 3C_1$ for characteristic three, and also assume, using Karatsuba's technique, that $M_2 = 3M_1$ and $M_3 = 6M_1$.

Note that the HCTBE and FDDE algorithms can easily be modified to work with window NAF techniques. In particular, we assume that the width-w radix-2 and radix-3 NAF representation of the exponent e are used in for the characteristic-two and the characteristic-three cases, respectively. Note that width-w radix-2 and radix-3 NAF representations of e contain on average $2\log_2 e/(w+1)$ and $2\log_3 e/(2w+1)$ nonzero digits, respectively; see for example [86].

The estimated costs of the exponentiation algorithms are presented in Table 4.1. In our analysis, we ignore the compression/decompression costs and also the precomputation costs required for window NAF methods as they are negligible comparing to the overall cost of algorithms.

Assuming that S_1 and C_1 are essentially free in characteristic-two and characteristic-three fields, respectively, and setting $w = 3$, we can estimate the cost of FDDE as

Table 4.1: Comparison of exponentiation algorithms for factor-4 and factor-6 compression in the case of a general base. The exponent is e.

Algorithms	Main Loop
Characteristic-two fields	
Algorithm 3.1	$(3.19M_1)\log_2 e$
FDDE	$(4S_1 + \frac{9}{(w+1)}M_1)\log_2 e$
HCTBE	$(4S_1 + \frac{6}{(w+1)}M_1)\log_2 e$
Characteristic-three fields	
XTR_3 in [78]	$(3M_1)\log_2 e$
FDDE	$(6C_1 + \frac{36}{(2w+1)}M_1)\log_3 e$
HCTBE	$(6C_1 + \frac{24}{(2w+1)}M_1)\log_3 e$

$(2.25M_1)\log_2 e$, and the cost of HCTBE as $(1.5M_1)\log_2 e$ in characteristic-two fields. Similarly, the cost of FDDE and HCTBE in characteristic-three fields can be approximated as $(3.24M_1)\log_2 e$ and $(2.16M_1)\log_2 e$, respectively.

Therefore, if we require that the input to an exponentiation algorithm and the output of the algorithm are the compressed representation of g and g^e, it seems best to compress g to $\mathcal{C}(g)$ by a factor of 4 or 6, and to use the HCTBE algorithms to compute the factor-4 or factor-6 compressed representation $\mathcal{C}(g^e)$ of g^e. It also seems that the HCTBE algorithms outperform the fastest previously-known exponentiation algorithms in G_ℓ. The reason is that compression/decompression costs in the HCTBE algorithms are negligible and that each multiplication step in the HCTBE algorithm in characteristic-two requires $6M_1$ whereas it would require $9M_1$ in a conventional exponentiation algorithm adapting Karatsuba's method. Similarly, each multiplication step in the HCTBE algorithm in characteristic-three requires $12M_1$ whereas it would require $18M_1$ in a conventional exponentiation algorithm adapting Karatsuba's method (see also Remark 4.6.3).

To be more concrete, we list the expected running times of the six exponentiation algorithms in a particular setting in Table 4.2 based on the estimates given in Table 4.1. For the 128-bit security level, in the characteristic-two case we let $q = 2^{1223}$ and $t = 2^{612}$. Then $q + 1 + t = 5\ell$ where ℓ is a 1221-bit prime. We will ignore the cost S_1. In the characteristic-three case, we let $q = 3^{509}$ and $t = 3^{255}$. Then $q + 1 - t = 7\ell$ where ℓ is an 804-bit prime. We will ignore the cost C_1. In both cases, we choose $w = 3$.

Table 4.2: Comparison of exponentiation algorithms for factor-4 and factor-6 compression in the case of a general base at the 128-bit security level. The exponent is an 1221-bit integer in the characteristic-two case, and an 804-bit integer in the characteristic-three case.

Algorithms	Main Loop
A characteristic-two field	
Algorithm 3.1	$3895M_1$
FDDE	$2747M_1$
HCTBE	$1831M_1$
A characteristic-three field	
XTR_3 in [78]	$2412M_1$
FDDE	$2609M_1$
HCTBE	$1739M_1$

4.8 Concluding remarks

We showed that by building on torus-based compression techniques, it is possible to compress elements in G_ℓ by a factor of 4 when $|G_\ell| = \ell = q + 1 \pm t$, $q = 2^m$ and $t = \sqrt{2q}$; and by a factor of 6 when $|G_\ell| = \ell = q + 1 - t$, $q = 3^m$ and $t = \sqrt{3q}$. Our methods achieve the best possible compression ratio in G_ℓ, and moreover have the feature that the compression and decompression maps are computable at a negligible cost. We discussed several exponentiation algorithms and, in particular, showed that HCTBE outperforms the fastest exponentiation algorithms in both the characteristic-two and the characteristic-three cases.

We believe that our techniques can be adapted for groups G_ℓ where $\ell = q + 1 + t$ and $q = 3^m$.

Our compression method compresses $g \in G_\ell$ to an element $\mathcal{C}(g)$ in $\mathbb{F}_q$. However, given $\mathcal{C}(g)$ and $e \in \mathbb{Z}$, all the exponentiation algorithms to compute $\mathcal{C}(g^e)$ first decompresses $\mathcal{C}(g)$ (at least partially), and then exponentiate. It is natural to ask if one can devise a multiplication formula for $g, h \in G_\ell$ which computes $\mathcal{C}(g) * \mathcal{C}(h) = \mathcal{C}(gh)$ directly in $\mathbb{F}_q$

Chapter 5

Validation in (Hyper)elliptic Curve Cryptosystems

In this chapter we extend the notion of an invalid-curve attack from elliptic curves to genus-2 hyperelliptic curves. We also show that invalid singular (hyper)elliptic curves can be used in mounting invalid-curve attacks on (hyper)elliptic curve cryptosystems, and make quantitative estimates of the practicality of these attacks. We thereby show that proper key validation is necessary even in cryptosystems based on hyperelliptic curves. As a byproduct, we enumerate the isomorphism classes of genus-g hyperelliptic curves over a finite field by a new counting argument that is simpler than the previous methods.

The remainder of this chapter is organized as follows. After laying some of the mathematical groundwork in Section 5.1, we present in Section 5.2 our derivation of the number of genus-g hyperelliptic curves over a finite field. In Section 5.3, we extend the notion of an invalid curve from elliptic curves to genus-2 curves. We also present the notion of an invalid singular curve, and enumerate the invalid elliptic and genus-2 singular curves. Our invalid-curve attacks are demonstrated and analyzed in Section 5.4; we conclude in Section 5.5.

The results of this chapter are joint work with B. Ustaoglu and are to appear in [51].

5.1 Preliminaries and previous results

Notation. The operator $[x^i]$ denotes the coefficient extraction operator when x is an indeterminate. For indeterminate x and polynomial $f(x)$ we adopt the convention $[x^i]f(x) = f_i$. The set of monic polynomials of degree d over a finite field $\mathbb{F}_q$ is denoted by $\mathcal{P}^d$; the subset of polynomials with at least one repeated root will be denoted by $\tilde{\mathcal{P}}^d$. Let

$\tilde{f} = \tilde{f}_{d-1}, \tilde{f}_{d-2}, \ldots, \tilde{f}_{d-i}$ be an ordered sequence where each $\tilde{f}_j \in \mathbb{F}_q$. Then

$$\mathcal{P}^d_{\tilde{f}} := \{x^d + \tilde{f}_{d-1}x^{d-1} + \cdots + \tilde{f}_{d-i}x^{d-i} + f_{d-i-1}x^{d-i-1} + \cdots + f_0 \mid f_{d-i-1}, \ldots, f_0 \in \mathbb{F}_q\}.$$

For example, $\mathcal{P}^2_{-3}$ denotes the set of polynomials of the form $x^2 - 3x + f_0$ where $f_0 \in \mathbb{F}_q$.

A *hyperelliptic curve* $\mathcal{H}$ of *genus-g* over a finite field $\mathbb{F}_q$ is defined by a non-singular *Weierstrass equation*

$$\mathcal{H}: \ y^2 + H(x)y = F(x),$$

where $F, H \in \mathbb{F}_q[x]$, F is monic, $\deg(F) = 2g + 1$, and $\deg(H) \leq g$. The *Jacobian* $J_{\mathcal{H}}(\mathbb{F}_q)$ of $\mathcal{H}$ over $\mathbb{F}_q$ is the quotient group of the *degree zero divisors* defined over $\mathbb{F}_q$ by the group of *principal divisors* defined over $\mathbb{F}_q$. The divisor classes $\bar{D} \in J_{\mathcal{H}}(\mathbb{F}_q)$ are in one-to-one correspondence with the pairs of polynomials (u, v) with $u, v \in \mathbb{F}_q[x]$, $\deg(v) < \deg(u) \leq g$, u monic, and $u \mid (v^2 + Hv - F)$; we write $\bar{D} = [u, v]$. $J_{\mathcal{H}}(\mathbb{F}_q)$ is a finite abelian group with $|J_{\mathcal{H}}(\mathbb{F}_q)| \in [(\sqrt{q} - 1)^{2g}, (\sqrt{q} + 1)^{2g}]$ [92]. Given two divisor classes $\bar{D}_1 = [u_1, v_1]$ and $\bar{D}_2 = [u_2, v_2] \in J_{\mathcal{H}}(\mathbb{F}_q)$, Cantor's algorithm [14] can be used to find the unique divisor $\bar{D} = [u, v]$ such that $\bar{D} = \bar{D}_1 + \bar{D}_2$.

If $\text{char}(\mathbb{F}_q) \notin \{2, 2g + 1\}$ then the same curve $\mathcal{H}$ (up to isomorphism) can be given by the equation

$$\mathcal{H}: y^2 = f(x) = x^{2g+1} + \sum_{i=0}^{2g-1} f_i x^i. \tag{5.1.1}$$

The non-singularity requirement on the equation of $\mathcal{H}$ means that f has no repeated roots, in which case $\mathcal{H}$ is said to be *non-singular*. If an equation (5.1.1) has the property that f has x_0 as a repeated root, we call $\mathcal{H}$ a *singular curve*, and (x_0, y_0), where $y_0^2 = f(x_0)$, a singular point on $\mathcal{H}$.

The remainder of this work assumes that a hyperelliptic curve $\mathcal{H}$ of genus-g over a finite field $\mathbb{F}_q$ is given via (5.1.1). The set of all (non-singular) genus-g hyperelliptic curves over $\mathbb{F}_q$ will be denoted by $\mathcal{H}^*$. When a hyperelliptic curve $\mathcal{H}$ is defined over $\mathbb{F}_q$ we will abbreviate $J_{\mathcal{H}}(\mathbb{F}_q)$ to $J_{\mathcal{H}}$.

5.2 The number of genus-g hyperelliptic curves

In this section we estimate the number of non-isomorphic genus-g hyperelliptic curves given by (5.1.1). First we need the following formula for the number of monic polynomials with no repeated roots and fixed second leading coefficient.

Theorem 5.2.1. *Let $g \geq 1$ be an integer, $\mathbb{F}_q$ be a finite field, and $f_{2g} \in \mathbb{F}_q$. Then*

$$|\mathcal{P}^{2g+1}_{f_{2g}} \setminus \tilde{\mathcal{P}}^{2g+1}_{f_{2g}}| = q^{2g} - q^{2g-1}.$$

Proof. Let $\bar{\mathbb{F}}_q$ denote the algebraic closure of $\mathbb{F}_q$. The argument proceeds by induction on g.

For $g = 1$, consider $f(x) = x^3 + f_2x^2 + f_1x + f_0 \in \mathcal{P}^3_{f_2}$. Since f has degree three, it can have at most one repeated root, say α. If $\alpha \in \bar{\mathbb{F}}_q \setminus \mathbb{F}_q$ then the conjugates of α are also repeated roots of f contradicting the fact that f has at most one repeated root. Therefore $\alpha \in \mathbb{F}_q$. Factoring f gives $f(x) = (x-\alpha)^2(x-\beta)$, where $\beta = -f_2 - 2\alpha$. Hence, the only degree of freedom is α and so $|\tilde{\mathcal{P}}^3_{f_2}| = q$. And, since $|\mathcal{P}^3_{f_2}| = q^2$, we have $|\mathcal{P}^3_{f_2} \setminus \tilde{\mathcal{P}}^3_{f_2}| = q^2 - q$.

Assume the result holds for all integers $1, 2, \ldots, (g-1)$, where $g \geq 2$. We will show the result holds for g. Let $f(x) = x^{2g+1} + \sum_{j=0}^{2g} f_j x^j \in \mathcal{P}^{2g+1}_{f_{2g}}$. To each repeated root α of f with multiplicity $k \geq 2$, we associate $\lfloor k/2 \rfloor$ pairs (α, α); we call each such pair a *paired repeated root* of f corresponding to α. Note that f can have at most g paired repeated roots. Since $f \in \mathbb{F}_q[x]$, if f has exactly i paired repeated roots then it can be written as $f = a^2b$, where $a, b \in \mathbb{F}_q[x]$,

$$\begin{aligned} a(x) &= x^i + a_{i-1}x^{i-1} + \cdots + a_0, \\ b(x) &= x^{2(g-i)+1} + \beta x^{2(g-i)} + b_{2(g-i)-1}x^{2(g-i)-1} + \cdots + b_0, \end{aligned}$$

and b has no repeated roots in $\bar{\mathbb{F}}_q$. Since the coefficient β in $b(x)$ satisfies $\beta + [x^{2i-1}]a(x)^2 = f_{2g}$, β is determined by a and f_{2g}. For a fixed $i \in [1, g-1]$ the number of polynomials a is q^i. By induction, the number of polynomials b of degree $2(g-i)+1$ that have no repeated roots and have fixed β is $q^{2(g-i)} - q^{2(g-i)-1}$. Therefore, the number of polynomials f with exactly i paired repeated roots is $q^i \cdot (q^{2(g-i)} - q^{2(g-i)-1}) = q^{2g-i} - q^{2g-i-1}$. For $i = g$ the polynomial f factors as $a^2(x-\beta)$ and as before β is determined by a and f_{2g}. Then the number of polynomials f with exactly g paired repeated roots equals the number of choices for a, which is q^g.

Hence, the number of polynomials f with at least one paired repeated root is

$$|\tilde{\mathcal{P}}^{2g+1}_{f_{2g}}| = q^g + \sum_{i=1}^{g-1}(q^{2g-i} - q^{2g-i-1}) = q^g + q^{2g-1} + \sum_{i=2}^{g-1} q^{2g-i} - \sum_{i=1}^{g-1} q^{2g-i-1} = q^{2g-1}.$$

Finally, since $|\mathcal{P}^{2g+1}_{f_{2g}}| = q^{2g}$, we have

$$|\mathcal{P}^{2g+1}_{f_{2g}} \setminus \tilde{\mathcal{P}}^{2g+1}_{f_{2g}}| = |\mathcal{P}^{2g+1}_{f_{2g}}| - |\tilde{\mathcal{P}}^{2g+1}_{f_{2g}}| = q^{2g} - q^{2g-1},$$

which completes the argument. □

Setting $f_{2g} = 0$ in Theorem 5.2.1 determines the number of polynomials that define a genus-g hyperelliptic curve over $\mathbb{F}_q$, where $\mathrm{char}(\mathbb{F}_q) \notin \{2, 2g+1\}$. However, such curves can have more than one representation, and we do not wish to distinguish between isomorphic curves. The following result, due to Lockhart [63], gives a one-to-one correspondence between isomorphism classes of curves and equivalence classes of Weierstrass equations.

Theorem 5.2.2. *[63, Proposition 1.2] If $\mathcal{H}_1$ and $\mathcal{H}_2$ are two genus-g hyperelliptic curves defined over $\mathbb{F}_q$, then $\mathcal{H}_1$ and $\mathcal{H}_2$ are isomorphic over $\mathbb{F}_q$ if and only if there exists $\alpha \in \mathbb{F}_q^*$, $\beta \in \mathbb{F}_q$, and $t \in \mathbb{F}_q[x]$ with $deg(t) \leq g$, such that the change of coordinates $(x, y) \rightarrow (\alpha^2 x + \beta, \alpha^{2g+1} y + t(x))$, transforms the equation of $\mathcal{H}_1$ to the equation of $\mathcal{H}_2$.*

Invalid-curve attacks are based on the curve representation (and the explicit group law). Hence we are interested in isomorphisms that preserve the representation of curves. Lockhart's theorem can be specialized to suit our needs as follows.

Suppose that $\mathrm{char}(\mathbb{F}_q)$ is odd and $\mathrm{char}(\mathbb{F}_q) \nmid (2g+1)$. Let $\mathcal{H}$ be a genus-g hyperelliptic curve over $\mathbb{F}_q$, and let $\tau : (x, y) \rightarrow (\alpha^2 x + \beta, \alpha^{2g+1} y + t(x))$ be an isomorphism to another genus-g hyperelliptic curve over $\mathbb{F}_q$. We claim that if τ preserves the form of equation (5.1.1), it must be the case that $\beta = 0$ and $t(x) = 0$. Indeed, if $t(x) \neq 0$ then applying τ to $\mathcal{H}$ will result in a linear term in y, which is not present in (5.1.1). Now, applying the transformation $(x, y) \rightarrow (\alpha^2 x + \beta, \alpha^{2g+1} y)$ to (5.1.1), the coefficient of x^{2g} is $(2g+1)\beta\alpha^{4g}$ which has to be zero as in (5.1.1). Since $\alpha \neq 0$ and $\mathrm{char}(\mathbb{F}_q) \nmid (2g+1)$ it must be the case that $\beta = 0$. Therefore, the class of transformations that correspond to isomorphisms of a hyperelliptic curve that preserve (5.1.1) are of the form $(x, y) \rightarrow (\alpha^2 x, \alpha^{2g+1} y)$, where $\alpha \in \mathbb{F}_q^*$. From now on, when we talk about isomorphisms we will identify them with the element α.

In Theorem 5.2.3 we estimate the number of isomorphism classes of genus-g hyperelliptic curves over finite fields $\mathbb{F}_q$ of odd characteristic not dividing $2g+1$. Note that by Theorem 5.2.1, the number of polynomials $f \in \mathbb{F}_q[x]$ that give rise to a genus-g hyperelliptic curve is $q^{2g} - q^{2g-1}$, and since the number of isomorphisms (nonzero field elements) is $q-1$ we expect that the number of non-isomorphic curves to be about q^{2g-1}. A slightly stronger result is proved independently in [71, Theorem 3.3]. The proof given here is simpler and is presented for completeness.

Theorem 5.2.3. *Let g be fixed. Let $\mathbb{F}_q$ be a finite field of odd characteristic p, and suppose that $p \nmid (2g+1)$ and $q > 4g+2$. Then the number of non-isomorphic genus-g hyperelliptic curves over $\mathbb{F}_q$ is*

$$N_g(q) = 2q^{2g-1} + \mathcal{O}(gq^{2g-2}).$$

Proof. Let $f \in \mathcal{P}_0^{2g+1} \setminus \tilde{\mathcal{P}}_0^{2g+1}$. Define $z_i(f) = 0$ if $f_i = 0$, and $z_i(f) = 1$ otherwise. We will abbreviate $z_i(f)$ to z_i in case f is clear from context. We call the sequence $z(f) = (z_0, z_1, \ldots, z_{2g-1})$ the *characteristic* sequence of f. We will use $ab\tilde{z}$ to denote the sequence z where $z_0 = a$, $z_1 = b$, and the remaining entries are given by $\tilde{z}$. Let $\mathcal{H}_{ab}^z$ be the set of polynomials f with characteristic sequence z such that $(z_0, z_1) = (a, b)$. Let $|z|$ denote the number of nonzero entries in a sequence z. Then $|\mathcal{H}_{10}^z| \leq (q-1)^{|z|}$ and $|\mathcal{H}_{01}^z| \leq (q-1)^{|z|}$.

An isomorphism α acting on the curve $y^2 = f(x)$ preserves the characteristic sequence z. Indeed, applying an isomorphism $\alpha : (x, y) \to (\alpha^2 x, \alpha^{2g+1} y)$ to (5.1.1) results in

$$\alpha^{4g+2} y^2 = \alpha^{4g+2} x^{2g+1} + \alpha^{4g-2} f_{2g-1} x^{2g-1} + \cdots + \alpha^2 f_1 x + f_0,$$

which can be rewritten as

$$y^2 = x^{2g+1} + \alpha^{-4} f_{2g-1} x^{2g-1} + \cdots + \alpha^{-4g} f_1 x + \alpha^{-4g-2} f_0.$$

Thus the curves $y^2 = f(x)$ for $f \in \mathcal{H}_{ab}^z$ have the same automorphism group; we denote this group by $\mathtt{Aut}\mathcal{H}_{ab}^z$. Observe also that if the mapping α is an automorphism then the order of α in $\mathbb{F}_q$ must divide $4g + 2$. Since $q > 4g + 2$, $|\mathtt{Aut}\mathcal{H}_{ab}^z|$ is no larger than $4g + 2$ which yields

$$\begin{aligned}
\sum_z |\mathcal{H}_{01}^z|(|\mathtt{Aut}\mathcal{H}_{01}^z| - 2) &\leq 4g \sum_z |\mathcal{H}_{01}^z| \leq 4g \sum_z (q-1)^{|z|} \\
&= 4g \sum_{|z|=1}^{2g-1} \binom{2g-2}{|z|-1} (q-1)^{|z|} \\
&= 4g(q-1) \sum_{i=1}^{2g-1} \binom{2g-2}{i-1} (q-1)^{i-1} \\
&= 4g(q-1) \sum_{i=0}^{2g-2} \binom{2g-2}{i} (q-1)^{i} \\
&= 4g(q-1) q^{2g-2} \\
&= \mathcal{O}(g q^{2g-1}).
\end{aligned}$$

Note that in the above equations, the sequence z has its first two coordinates fixed, and therefore the remaining $|z| - 1$ nonzero entries are chosen from $2g - 2$ possible indices. Similarly, we have

$$\sum_z |\mathcal{H}_{10}^z|(|\mathtt{Aut}\mathcal{H}_{10}^z| - 2) = \mathcal{O}(g q^{2g-1}).$$

If both z_0 and z_1 are equal to zero, then 0 is a repeated root of f. So if f has no repeated roots then at least one of z_0 or z_1 is nonzero. By Theorem 5.2.1 we have

$$\sum_z (|\mathcal{H}_{11}^z| + |\mathcal{H}_{01}^z| + |\mathcal{H}_{10}^z|) = q^{2g} - q^{2g-1}.$$

An automorphism α that fixes f when f_1 and f_0 are simultaneously nonzero must satisfy

$\alpha^2 = 1$. There are two such automorphisms, so $|\mathtt{Aut}\mathcal{H}^z_{11}| = 2$. Therefore,

$$\begin{aligned}
N_g(q) &= \sum_z \frac{|\mathcal{H}^z_{ab}|}{[\mathbb{F}_q^* : \mathtt{Aut}\mathcal{H}^z_{ab}]} = \sum_z \frac{|\mathcal{H}^z_{ab}||\mathtt{Aut}\mathcal{H}^z_{ab}|}{|\mathbb{F}_q^*|} = \sum_z \frac{|\mathcal{H}^z_{ab}||\mathtt{Aut}\mathcal{H}^z_{ab}|}{q-1} \\
&= \sum_{01\tilde{z}} \frac{|\mathcal{H}^z_{01}||\mathtt{Aut}\mathcal{H}^z_{01}|}{q-1} + \sum_{10\tilde{z}} \frac{|\mathcal{H}^z_{10}||\mathtt{Aut}\mathcal{H}^z_{10}|}{q-1} + \sum_{11\tilde{z}} \frac{|\mathcal{H}^z_{11}||\mathtt{Aut}\mathcal{H}^z_{11}|}{q-1} \\
&= \frac{1}{q-1}\left(\sum_{01\tilde{z}} |\mathcal{H}^z_{01}||\mathtt{Aut}\mathcal{H}^z_{01}| + \sum_{10\tilde{z}} |\mathcal{H}^z_{10}||\mathtt{Aut}\mathcal{H}^z_{10}| + \sum_{11\tilde{z}} 2|\mathcal{H}^z_{11}|\right) \\
&= \frac{1}{q-1}\left(\sum_{01\tilde{z}} |\mathcal{H}^z_{01}||\mathtt{Aut}\mathcal{H}^z_{01}| + \sum_{10\tilde{z}} |\mathcal{H}^z_{10}||\mathtt{Aut}\mathcal{H}^z_{10}| + 2\left(q^{2g} - q^{2g-1} - \sum_{01\tilde{z}} |\mathcal{H}^z_{01}| - \sum_{10\tilde{z}} |\mathcal{H}^z_{10}|\right)\right) \\
&= \frac{1}{q-1}\left(\sum_{01\tilde{z}} |\mathcal{H}^z_{01}|\left(|\mathtt{Aut}\mathcal{H}^z_{01}| - 2\right) + \sum_{10\tilde{z}} |\mathcal{H}^z_{10}|\left(|\mathtt{Aut}\mathcal{H}^z_{10}| - 2\right) + 2\left(q^{2g} - q^{2g-1}\right)\right) \\
&= \frac{1}{q-1}\left(\mathcal{O}(gq^{2g-1}) + \mathcal{O}(gq^{2g-1}) + 2q^{2g} - 2q^{2g-1}\right) \\
&= \frac{1}{q-1}\left(2q^{2g} + \mathcal{O}(gq^{2g-1})\right) \\
&= 2q^{2g-1} + \mathcal{O}(gq^{2g-2}),
\end{aligned}$$

as required. □

5.3 Invalid and singular curves

We now extend the notion of invalid elliptic curves, proposed by Antipa et al. [4], to genus-2 curves. We emphasize that invalid curves are defined with respect to a specific curve representation and explicit formulae for the group law. That is, given a curve representation and formulae for the group operations that do not make use of a specific coefficient in the selected curve representation, one can define an invalid curve with respect to that representation-formulae pair. If the explicit formulae utilize all the coefficients in the curve representation then invalid curves in this context do not exist. However, for curves of genus-1 and genus-2, which are widely considered for cryptographic applications, the notion of invalid curves is indeed relevant and important.

In the genus-1 setting, we use the affine formulae for the group law as described in [5, Section 13.2.1], and refer to these formulae as $\mathcal{F}_{1a}$ throughout this chapter. The explicit computations in $\mathcal{F}_{1a}$ require only the coefficient f_1. In our definition of invalid elliptic curves, we include singular elliptic curves as well.

Definition 5.3.1. Let $\mathcal{E}$ be an elliptic curve defined over $\mathbb{F}_q$ with equation

$$\mathcal{E} : y^2 = x^3 + f_1 x + f_0.$$

An *invalid curve* relative to $\mathcal{E}$ and $\mathcal{F}_{1a}$ is an elliptic curve over $\mathbb{F}_q$ with equation

$$\mathcal{IE} : y^2 = x^3 + f_1 x + \tilde{f}_0,$$

where $\tilde{f}_0 \neq f_0$ and $\mathcal{IE}$ is not isomorphic to $\mathcal{E}$. In addition, if the polynomial $\tilde{f}(x) = x^3 + f_1 x + \tilde{f}_0$ has a repeated root then $\mathcal{IE}$ is called an *invalid singular curve* relative to $\mathcal{E}$ and $\mathcal{F}_{1a}$.

In the genus-2 setting, we will use the affine formulae for the group law as described in [5, Section 14.3.2], and refer to these formulae as $\mathcal{F}_{2a}$ throughout this chapter. The formulae $\mathcal{F}_{2a}$ depends only on f_2 and f_3. In our definition of invalid hyperelliptic curves, we include singular hyperelliptic curves as well.

Definition 5.3.2. Let $\mathcal{H}$ be a genus-2 hyperelliptic curve defined over $\mathbb{F}_q$ with equation

$$\mathcal{H} : y^2 = x^5 + f_3 x^3 + f_2 x^2 + f_1 x + f_0.$$

An *invalid curve* relative to $\mathcal{H}$ and $\mathcal{F}_{2a}$ is a hyperelliptic curve over $\mathbb{F}_q$ with equation

$$\mathcal{IH} : y^2 = x^5 + f_3 x^3 + f_2 x^2 + \tilde{f}_1 x + \tilde{f}_0,$$

where $(\tilde{f}_1, \tilde{f}_0) \neq (f_1, f_0)$ and $\mathcal{IH}$ is not isomorphic to $\mathcal{H}$. In addition, if the polynomial $\tilde{f}(x) = x^5 + f_3 x^3 + f_2 x^2 + \tilde{f}_1 x + \tilde{f}_0$ has a repeated root then $\mathcal{IH}$ is called an *invalid singular curve* relative to $\mathcal{H}$ and $\mathcal{F}_{2a}$.

Invalid singular curves are very interesting in the genus-1 case. If $\mathcal{SE}$ is an invalid singular curve over $\mathbb{F}_q$ relative to the elliptic curve $\mathcal{E}$ over $\mathbb{F}_q$ and $\mathcal{F}_{1a}$, then $\mathcal{SE}$ has exactly one singular point $P = (x_0, y_0)$. Applying the isomorphism $(x, y) \to (x + x_0, y + y_0)$ to $\mathcal{SE}$, we can assume that $\mathcal{SE}$ is given by the equation

$$\mathcal{SE} : y^2 = x^3 + a_2 x^2,\ a_2 \in \mathbb{F}_q,$$

and $P = (0, 0)$ is the singular point of $\mathcal{SE}$. Now, let $y^2 - a_2 x^2 = (y - \alpha x)(y - \beta x)$ where $\alpha, \beta \in \bar{\mathbb{F}}_q$. If $a_2 = 0$ then $\alpha = \beta = 0$, and P is called a *cusp* singularity of $\mathcal{SE}$. If $a_2 \neq 0$ then $\alpha = -\beta$, and $\alpha^2 = a_2$; P is called a *node* singularity of $\mathcal{SE}$. In this case, $\alpha, \beta \in \mathbb{F}_q$ if a_2 is a quadratic residue in $\mathbb{F}_q$; and $\alpha, \beta \in \mathbb{F}_{q^2} \setminus \mathbb{F}_q$, otherwise. It is well known that the set $\mathcal{SE}_{ns}(\mathbb{F}_q)$ of non-singular $\mathbb{F}_q$-points on $\mathcal{SE}$ together with the point at infinity forms a group and in fact the group law $\mathcal{F}_{1a}$ for $\mathcal{E}$ is also the group law for $\mathcal{SE}$. Moreover, if P is a cusp singularity of $\mathcal{SE}$ then $\mathcal{SE}_{ns}(\mathbb{F}_q)$ is isomorphic to the additive group of $\mathbb{F}_q$. If

P is a node singularity of $\mathcal{SE}$ and $\alpha \in \mathbb{F}_q$ then $\mathcal{SE}_{ns}(\mathbb{F}_q)$ is isomorphic to $\mathbb{F}_q^*$; and if P is a node singularity of $\mathcal{SE}$ and $\alpha \in \mathbb{F}_{q^2} \setminus \mathbb{F}_q$ then $\mathcal{SE}_{ns}(\mathbb{F}_q)$ is isomorphic to the order-$(q+1)$ multiplicative subgroup of $\mathbb{F}_{q^2}^*$. In all cases, the isomorphisms and their inverses are efficiently computable (see [42, Section 7.2] for more details). The key point in applying singular invalid-curve attacks is that the group law for non-singular elliptic curves can readily be used for the non-singular part of singular elliptic curves.

The next theorem assumes the setting in Definition 5.3.1 and establishes the existence and the number of invalid singular elliptic curves.

Theorem 5.3.3. *Over $\mathbb{F}_q$, where $gcd(q, 6) = 1$, the number of invalid singular curves relative to $y^2 = x^3 + f_1x + f_0$ and $\mathcal{F}_{1a}$ is*

(i) 1, *if $f_1 = 0$;*

(ii) 2, *if $-\frac{f_1}{3}$ is a quadratic residue in $\mathbb{F}_q$.*

(iii) 0, *if $-\frac{f_1}{3}$ is a quadratic non-residue in $\mathbb{F}_q$;*

Proof. Let $f_1 \in \mathbb{F}_q$ be fixed and consider the set of polynomials

$$\mathcal{P}^3_{0,f_1} = \{P_{f_0'}(x) = x^3 + f_1x + f_0' : f_0' \in \mathbb{F}_q\}.$$

If $P_{f_0'}(x)$ has a repeated root in $\bar{\mathbb{F}}_q$ then we must have

$$P_{f_0'}(x) = x^3 + f_1x + f_0' = (x + a)^2(x + k_0),\ a, k_0 \in \mathbb{F}_q,$$

or equivalently,

$$k_0 = -2a \tag{5.3.1}$$
$$f_1 = -3a^2 \tag{5.3.2}$$
$$f_0' = -2a^3. \tag{5.3.3}$$

For a fixed f_1, we consider the solutions (a, k_0, f_0') to (5.3.1)–(5.3.3).

Case (i)**.** If $f_1 = 0$ then $(0, 0, 0)$ is the only solution to (5.3.1)–(5.3.3) and so $\mathcal{P}^3_{0,f_1}$ has exactly one polynomial that has a repeated root, namely $P(x) = x^3$. In this case, the curve defined by $y^2 = P(x)$ has a cusp singularity at $S = (0, 0)$.

Case (ii)**.** If $-f_1/3 = a_1^2$ for some $a_1 \in \mathbb{F}_q^*$ then $(a_1, -2a_1, -2a_1^3)$ and $(-a_1, 2a_1, 2a_1^3)$ are the only two solutions to (5.3.1)–(5.3.3). Hence, $\mathcal{P}^3_{0,f_1}$ has exactly two polynomials with repeated roots: $P(x) = x^3 + f_1x - 2a_1^3$ in which case the curve defined by $y^2 = P(x)$ has a node singularity at $S = (-a_1, 0)$; and $P(x) = x^3 + f_1x + 2a_1^3$ in which case the curve defined by $y^2 = P(x)$ has a node singularity at $S = (a_1, 0)$.

Case (*iii*)**.** If $-f_1/3$ is a quadratic non-residue in $\mathbb{F}_q$ then the system defined by (5.3.1)–(5.3.3) has no solutions and so $\mathcal{P}^3_{0,f_1}$ does not contain any polynomial with repeated roots. □

In the attacks described in Section 5.4, the adversary will need curves with small-order subgroups. In the genus-1 case the adversary has the ability to choose $\tilde{f}_0$, and in [4, Section 4.4] it was already argued that an adversary can efficiently find suitable invalid curves, essentially by picking curves at random. To extend that argument for genus-2 we need the following result.

Theorem 5.3.4. *Suppose that* $gcd(q, 30) = 1$*. The number of invalid singular curves relative to* $y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$ *and* $\mathcal{F}_{2a}$ *over* $\mathbb{F}_q$ *is between* $q - 3$ *and* $q + 3$*.*

Proof. Let $f_2, f_3 \in \mathbb{F}_q$ be fixed, and consider the set of polynomials

$$\mathcal{P}^5_{0,f_3,f_2} = \{P_{f_1',f_0'}(x) = x^5 + f_3x^3 + f_2x^2 + f_1'x + f_0' : \ f_1', f_0' \in \mathbb{F}_q\}.$$

If $P_{f_1',f_0'}(x)$ has repeated roots in $\bar{\mathbb{F}}_q$, then there are two (not mutually exclusive) possibilities:

Case 1. There exist $a, b, k_0 \in \mathbb{F}_q$ such that

$$P_{f_1',f_0'}(x) = x^5 + f_3x^3 + f_2x^2 + f_1'x + f_0' = (x^2 + ax + b)^2(x + k_0). \tag{5.3.4}$$

Comparing the coefficients of the same degree terms, we have

$$\begin{aligned} k_0 &= -2a && (5.3.5)\\ f_3 &= 2b - 3a^2 && (5.3.6)\\ f_2 &= -(2a^3 + 2ab) && (5.3.7)\\ f_1' &= b^2 - 4a^2b && (5.3.8)\\ f_0' &= -2b^2a. && (5.3.9) \end{aligned}$$

We obtain from (5.3.6) and (5.3.7) that a must satisfy $5a^3 + f_3a + f_2 = 0$. Moreover, since f_2 and f_3 are already fixed, any choice of a fixes k_0 and b by (5.3.5) and (5.3.6). Therefore, the number of polynomials $P_{f_1',f_0'}(x)$ of the form (5.3.4) is at most three.

Case 2. There exist $a, k_0, k_1, k_2 \in \mathbb{F}_q$ such that

$$P_{f_1',f_0'}(x) = x^5 + f_3x^3 + f_2x^2 + f_1'x + f_0' = (x + a)^2(x^3 + k_2x^2 + k_1x + k_0).$$

Comparing the coefficients of the same degree terms, we have

$$\begin{aligned} k_2 &= -2a && (5.3.10)\\ k_1 &= f_3 + 3a^2 && (5.3.11)\\ k_0 &= f_2 - 2a(f_3 + 2a^2) && (5.3.12)\\ f_1' &= 2af_2 - 3a^2 f_3 - 5a^4 && (5.3.13)\\ f_0' &= a^2 f_2 - 2a^3 f_3 - 4a^5. && (5.3.14) \end{aligned}$$

For a fixed pair $(f_2, f_3) \in \mathbb{F}_q \times \mathbb{F}_q$, we first count the number of solutions $(a, k_0, k_1, k_2, f_0', f_1')$ to (5.3.10)–(5.3.14). Note that if $a = 0$ then we must have $k_0 = f_2,\ k_1 = f_3,\ k_2 = f_0' = f_1' = 0$. On the other hand, if $a \neq 0$, we have $k_1 \neq f_3$ and $(k_1 - f_3)/3$ is a quadratic residue in $\mathbb{F}_q$ for exactly $(q-1)/2$ elements $k_1 \in \mathbb{F}_q$. For each such k_1, we obtain two solutions determined by setting a equal to the square roots of $(k_1 - f_3)/3$. That is, there are $1 + 2((q-1)/2) = q$ solutions in total and each solution $(a, k_0, k_1, k_2, f_0', f_1')$ leads to a polynomial $P_{f_1', f_0'}(x)$ which has either one or two paired repeated roots in $\mathbb{F}_q$. We note that different solutions may lead to the same polynomial $P_{f_1', f_0'}(x)$. It is easy to see that this occurs only if $P_{f_1', f_0'}(x)$ has exactly two paired repeated roots in $\mathbb{F}_q$ and in this case we show that, for a fixed $P_{f_1', f_0'}(x)$, there could exist at most two different solutions that yield this polynomial. The proof is as follows. Suppose there are three different solutions $S := (a, k_0, k_1, k_2, f_0', f_1')$, $\tilde{S} := (\tilde{a}, \tilde{k}_0, \tilde{k}_1, \tilde{k}_2, f_0', f_1')$ and $\hat{S} := (\hat{a}, \hat{k}_0, \hat{k}_1, \hat{k}_2, f_0', f_1')$ that yield the polynomial $P_{f_1', f_0'}(x)$. If $a = \tilde{a}$ then $S = \tilde{S}$ by (5.3.10), (5.3.11) and (5.3.12). Therefore, we will assume that $a, \tilde{a}$ and $\hat{a}$ are pairwise different; and in this case one can see that $(x + \tilde{a})^2 (x + \hat{a})^2 \mid (x^3 + k_2 x^2 + k_1 x + k_0)$, which is impossible.

We are now ready to prove the theorem. Suppose that the number of polynomials $P_{f_1', f_0'}(x)$ that have exactly two paired repeated roots both of which are in $\mathbb{F}_q$ is β, and the number of polynomials $P_{f_1', f_0'}(x)$ that have exactly two paired repeated roots both of which are in $\mathbb{F}_{q^2} \setminus \mathbb{F}_q$ is γ. Then, by our above argument, the number of solutions $(a, k_0, k_1, k_2, f_0', f_1')$ to (5.3.10)–(5.3.14), such that each solution leads to a polynomial $P_{f_1', f_0'}(x)$ that has exactly two paired repeated roots both of which are in $\mathbb{F}_q$, is at most 2β. Hence, there are at least $q - 2\beta$ polynomials $P_{f_1', f_0'}(x)$ having exactly one paired repeated root in $\mathbb{F}_q$, and there are at least $(q - 2\beta) + \beta + \gamma$ polynomials $P_{f_1', f_0'}(x)$ with at least one repeated root in $\bar{\mathbb{F}}_q$. By Case 1, $\beta + \gamma \leq 3$, and we can see that $q - \beta + \gamma \geq q - 3$.

From Case 1 and Case 2 there are at most $q + 3$ polynomials $P_{f_1', f_0'}(x)$ with at least one repeated root in $\bar{\mathbb{F}}_q$. □

Remark 5.3.5. Let $\mathcal{H} : y^2 = x^5 + f_3 x^3 + f_2 x^2 + f_1 x + f_0$ be a genus-2 hyperelliptic curve defined over $\mathbb{F}_q$. According to Theorem 5.3.4, there are at least $\chi = q^2 - q - 3$ curve equations (including the equation of $\mathcal{H}$) relative to $\mathcal{H}$ and $\mathcal{F}_{2a}$. We now argue that if $f_2 \neq 0$ or $f_3 \neq 0$ then at least $\frac{\chi}{6}$ of these (non-singular) curves are pairwise non-isomorphic. Let

$\mathcal{H}'$ and $\mathcal{H}''$ be two genus-2 hyperelliptic curves over $\mathbb{F}_q$ such that

$$\begin{aligned} \mathcal{H}' &: y^2 = x^5 + f_3x^3 + f_2x^2 + f_1'x + f_0' \\ \mathcal{H}'' &: y^2 = x^5 + f_3x^3 + f_2x^2 + f_1''x + f_0''. \end{aligned}$$

The curves $\mathcal{H}'$ and $\mathcal{H}''$ are isomorphic if there is an isomorphism α

$$\alpha : (x, y) \rightarrow (\alpha^2 x, \alpha^5 y)$$

that applied to $\mathcal{H}'$ gives the equation of $\mathcal{H}''$. Applying α to $\mathcal{H}'$ gives

$$\alpha\mathcal{H}' : y^2 = x^5 + \alpha^{-4}f_3x^3 + \alpha^{-6}f_2x^2 + \alpha^{-8}f_1'x + \alpha^{-10}f_0'.$$

If $f_2 \neq 0$ or $f_3 \neq 0$ then $\mathcal{H}'$ is isomorphic to $\mathcal{H}''$ only if $\alpha^4 = 1$ or $\alpha^6 = 1$. This proves that at least $\frac{\chi}{6}$ of the curves relative to $\mathcal{H}$ and $\mathcal{F}_{2a}$ are pairwise non-isomorphic. Hence there are at least $(\frac{\chi}{6} - 1)$ isomorphism classes of invalid curves relative to $\mathcal{H}$ and $\mathcal{F}_{2a}$ when $f_2 \neq 0$ or $f_3 \neq 0$. We are justified in omitting the case $f_2 = f_3 = 0$ in our argument since there are at most 10 isomorphism classes of such genus-2 hyperelliptic curves defined over $\mathbb{F}_q$; see [28].

5.4 Invalid-curve attacks

Suppose that a discrete logarithm cryptographic protocol requires party $\hat{B}$ to use his static (long-term) private key b by computing $\sigma = bP$ for some incoming $P \in J_{\mathcal{H}}$, where $\mathcal{H}$ is a genus-1 or genus-2 hyperelliptic curve and P has order n. If $\hat{B}$ does not verify that $P \in J_{\mathcal{H}}$ then an adversary $\mathcal{M}$ could launch an invalid-curve attack to learn b by selecting $P \in J_{\mathcal{I}}$ where $\mathcal{I}$ is an invalid curve or an invalid singular curve with respect to $\mathcal{H}$ (and either $\mathcal{F}_{1a}$ or $\mathcal{F}_{2a}$). Invalid-curve attacks come in two flavors.

In a *small subgroup attack* [62], $\mathcal{M}$ selects $\mathcal{I}$ so that $J_{\mathcal{I}}$ has an element P of small order r. The order r is small enough so that the discrete logarithm problem in $\langle P \rangle$ is feasible via exhaustive search; typically r is a small prime. Small subgroup attacks can be used in situations where $\mathcal{M}$ is able to obtain a quantity that is derived from bP, for example $k = H(bP)$ where H is a cryptographic hash function. In this case, $\mathcal{M}$ would compute $k' = H(iP)$ for all $i \in [0, r-1]$ until $k' = k$, after which $\mathcal{M}$ concludes that $b \equiv i \pmod{r}$. By repeating the procedure for different curves $\mathcal{I}$ (and different primes r) the value b can be recovered via the Chinese Remainder Theorem.

In a *large subgroup attack* [64, p.58], $\mathcal{M}$ selects an invalid curve $\mathcal{I}$ so that $J_{\mathcal{I}}$ has an element P of large order $t \approx n$ and so that the discrete logarithm problem in $\langle P \rangle$ can be efficiently solved; this is the case if t is smooth, or if there exists an efficiently computable

mapping from $\langle P \rangle$ to another group where efficient DLP algorithms are known. Large subgroup attacks can be used in situations where $\mathcal{M}$ is able to obtain the group element bP itself. In this case, $\mathcal{M}$ would compute $b \bmod t$, and thereafter efficiently determine b.

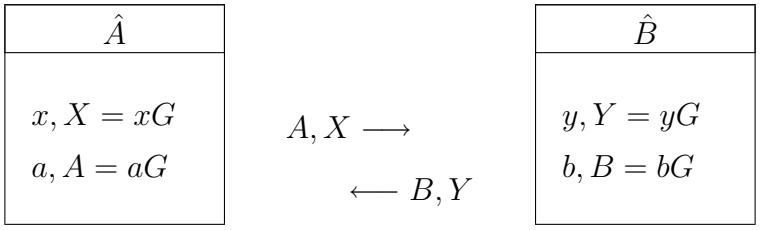
There are two main reasons that invalid-curve attacks can be used by $\mathcal{M}$ in the above setting. First, the representation of elements in the valid group $J_{\mathcal{H}}$ is the same as the representation of elements in the invalid group $J_{\mathcal{I}}$. Second, the implementation of the group operation in the valid group $J_{\mathcal{H}}$ is applicable to the invalid group $J_{\mathcal{I}}$ without any modification. We illustrate our invalid-curve attacks on two recently-proposed discrete logarithm protocols — the Twin Diffie-Hellman key agreement scheme and the XCR signature scheme — that are successful if public-key validation is not performed. We emphasize that our attacks do not illustrate any weaknesses in these protocols, but rather serve to emphasize the importance of public-key validation. More precisely, the attacks we describe do not require the adversary to tamper with hardware or modify registers; the attacks exploit only omission of public-key validation. Since validation in the case of curve-based cryptography does not require full exponentiation but only a constant number of field multiplications, it introduces a negligible efficiency overhead. Therefore validation should always be performed in conjunction with curve-based protocols.

5.4.1 Twin Diffie-Hellman (TDH)

Cash, Kiltz and Shoup [16, Section 4] proposed and analyzed a simple Diffie-Hellman type protocol depicted in Figure 5.1. The security of TDH relies on the twin Diffie-Hellman assumption which is equivalent to the computational Diffie-Hellman assumption. This is in contrast with many other key agreement protocols (e.g. [57, 87]), where security has only been proven with respect to the gap Diffie-Hellman assumption – this is the assumption that the computational Diffie-Hellman problem is hard even if the solver is given an oracle for the decisional Diffie-Hellman problem.

We extend the small subgroup attacks on the static Diffie-Hellman protocol to the TDH protocol in the genus-2 setting. We show how, even in the restricted security model used in [16], an adversary can successfully break the protocol should honest parties fail to obtain assurances that the static public keys of their peers were validated. This demonstrates the importance of requiring that all elements belong to the correct group.

Informally, the security model in [16] allows the adversary $\mathcal{M}$ to observe the interaction between honest parties, obtain session keys computed by honest parties, register corrupt parties with $\mathcal{M}$'s choice of static public key, and interact with honest parties on behalf of corrupted parties. The model did not (explicitly) require any checks on the static public keys chosen by the adversary when registering corrupt parties. The implication is that the adversary can register invalid static keys.



$$\kappa = H\left(\hat{A}, \hat{B}, \mathrm{CDH}(A, B), \mathrm{CDH}(A, Y), \mathrm{CDH}(X, B), \mathrm{CDH}(X, Y)\right)$$

Figure 5.1: The twin Diffie-Hellman protocol. The underlying group is $\langle G \rangle$. Party $\hat{A}$'s static key pairs are (A, a) and (X, x), while party $\hat{B}$'s static key pairs are (B, b) and (Y, y).

We now describe an attack that allows $\mathcal{M}$ to recover the static private key of an honest party. Suppose that the underlying group is a prime-order subgroup of $J_{\mathcal{H}}$, where $\mathcal{H}$ is the hyperelliptic curve defined by the polynomial $y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$ over $\mathbb{F}_q$. Suppose also that honest parties use group addition formula $\mathcal{F}_{2a}$ (recall that $\mathcal{F}_{2a}$ does not explicitly use the coefficients f_1 and f_0). In this case $\mathcal{M}$ chooses invalid curves $\mathcal{IH}_1$ and $\mathcal{IH}_2$ relative to $\mathcal{H}$ and $\mathcal{F}_{2a}$ such that the invalid curves have points of small orders u and v, respectively, with $\gcd(u, v) = 1$.

Using the notation in Figure 5.1, assume that $\hat{A}$ is corrupt and $\hat{B}$ is honest. Suppose that when $\mathcal{M}$ registers $\hat{A}$, $\mathcal{M}$ picked $X \in \mathcal{IH}_1$ of order u and $A \in \mathcal{IH}_2$ of order v. Then $\mathcal{M}$ initiates a session with $\hat{B}$ and obtains the session key $\kappa = H(\hat{A}, \hat{B}, A^b, A^y, X^b, X^y)$ that $\hat{B}$ computes. Now $\mathcal{M}$ computes

$$\kappa' = H(\hat{A}, \hat{B}, A^{i_1}, A^{i_2}, X^{i_3}, X^{i_4}),$$

where i_1, i_2 range over $\mathbb{Z}_v$ and i_3, i_4 range over $\mathbb{Z}_u$, until $\kappa' = \kappa$ in which case $\mathcal{M}$ learns that $b \equiv i_1 \pmod{v}$, $b \equiv i_3 \pmod{u}$, $y \equiv i_2 \pmod{v}$ and $y \equiv i_4 \pmod{u}$. For concreteness, suppose that $u, v \approx 2^{10}$. Note that $\mathcal{M}$ has to perform roughly 2^{40} steps before $\kappa' = \kappa$, which is a feasible amount of computation. After repeating the procedure for other orders (u, v), $\mathcal{M}$ can recover $\hat{B}$'s static private key (b, y).

Recall from Remark 5.3.5 that there are at least $(q^2 - q - 9)/6$ isomorphism classes of invalid curves relative to $\mathcal{H}$ and $\mathcal{F}_{2a}$, where $f_2 \neq 0$ or $f_3 \neq 0$ in the equation of $\mathcal{H}$. We can reasonably assume that the distribution of orders of these invalid curves follows the distribution of orders of all genus-2 curves over $\mathbb{F}_q$. Under this assumption, the invalid curves have group orders that are almost uniformly distributed over the Hasse interval $[(\sqrt{q}-1)^4, (\sqrt{q}+1)^4]$ (see [60, Proposition 1.9] and [61, Theorems 1.1 and 11.5] for results on the distribution of genus-1 and 2 curves over the Hasse interval). Thus, by randomly selecting invalid curves $\mathcal{H}'$, the adversary will quickly find one with group order divisible by a small prime. The bottleneck in the search is the time for computing the cardinality

of $J_{\mathcal{H}'}(\mathbb{F}_q)$. With current technology, computing the cardinality of $J_{\mathcal{H}'}(\mathbb{F}_q)$ is feasible for 80-bit fields $\mathbb{F}_q$ [33], but not quite feasible for 128-bit fields $\mathbb{F}_q$.

5.4.2 Singular elliptic curves

Krawczyk [55, Section 4.1 Definition 2] proposed the exponential challenge-response (XCR) signature scheme for a cryptographically strong group $\mathcal{G} = \langle G \rangle$ of prime-order n. In the XCR signature scheme the signer $\hat{B}$ with static key pair (B, b) signs a message for a verifier $\hat{A}$ who submits a challenge $X = xG$. The signature on a message m with a challenge X is (σ, Y), where $Y = yG$ is a random group element of $\hat{B}$'s choice and $\sigma = (y + H(Y, m)b)X$. $\hat{A}$ can verify the signature (σ, Y) using her knowledge of x via $\sigma = x(Y + H(Y, m)B)$. In the description of XCR in [55] there is an additional check which is not applicable for elliptic curves and hence is omitted here.

As before the goal of the adversary $\mathcal{M}$ is to learn the static private key b of the signer $\hat{B}$. The large subgroup attack assumes that the adversary can learn the ephemeral private key y that $\hat{B}$ chooses to sign a message m. The attack proceeds as follows. $\mathcal{M}$ selects an arbitrary message m and an invalid curve $\mathcal{H}'$ having an element X of large smooth order $t \approx n$. The signer $\hat{B}$ signs m and returns (σ, Y) to $\mathcal{M}$. The adversary learns the ephemeral private key y and can thereafter deduce $b \bmod t$ by using the Pohlig-Hellman algorithm to compute the logarithm of $H(Y, m)^{-1}(\sigma - yX) = bX$ to the base X; b can then be efficiently determined. Alternatively, $\mathcal{M}$ can mount a small subgroup attack. In that case, assuming that $\mathcal{M}$'s computational power is $O(2^{50})$, $\mathcal{M}$ will need to select invalid curves $\mathcal{H}'$ having elements X of 100-bit order t. To recover the secret b, $\mathcal{M}$ will need roughly $\lfloor 521/100 \rfloor = 5$ interactions with the signer.

For concreteness, suppose now that the signer uses an elliptic curve of the form $y^2 = x^3 - 3x + f_0$ over the prime field $\mathbb{F}_p$ where $p = 2^{521} - 1$. Such an elliptic curve has been specified by NIST in the FIPS 186-2 Standard [30]. Consider

$$\mathcal{E}_- : y^2 = x^3 - 3x + 2 = (x + 2)(x - 1)^2$$

and

$$\mathcal{E}_+ : y^2 = x^3 - 3x - 2 = (x - 2)(x + 1)^2$$

over $\mathbb{F}_p$, where $\gcd(p, 6) = 1$. According to Theorem 5.3.3*(ii)*, $\mathcal{E}_+$ and $\mathcal{E}_-$ are invalid singular curves relative to any curve $\mathcal{E}$ defined via $y^2 = x^3 - 3x + f_0$ over $\mathbb{F}_p$. In particular, $\mathcal{E}_+$ and $\mathcal{E}_-$ are invalid singular curves relative to all the five NIST curves defined over prime fields [30]. The sets of non-singular points on $\mathcal{E}_+$ and $\mathcal{E}_-$ over $\mathbb{F}_p$ (together with the point at infinity) form groups isomorphic to $\mathbb{F}_p^*$ that share the same group law with $\mathcal{E}$. Moreover, the isomorphisms are efficiently computable (see the discussion after Definition 5.3.2). Using $\mathcal{E}_+$, $\mathcal{M}$ can mount the following large subgroup attack. $\mathcal{M}$ picks a point X of order

$p-1$ on $\mathcal{E}_+$. As before $\mathcal{M}$ interacts with the signer and obtains a signature (σ, Y) on a message m. Suppose $\mathcal{M}$ is able to learn $\hat{B}$'s ephemeral private key y. Then $\mathcal{M}$ can map $H(Y,m)^{-1}(\sigma - yX)$ to $\mu \in \mathbb{F}_p^*$ and X to $g \in \mathbb{F}_p^*$, and then use subexponential discrete logarithm algorithms [38, 48] to compute $\log_g \mu$ thus obtaining $b \bmod (p-1)$. Note that the discrete logarithm algorithm of [48] has a first phase that can be precomputed before the attack is launched. Subsequently, the second phase quickly computes individual logarithms and can reuse the precomputations in multiple applications of the attack.

We note that in the case of the NIST prime $p = 2^{521} - 1$, discrete logarithms in $\mathbb{F}_p^*$ can be more efficiently computed using the Pohlig-Hellman algorithm since the largest prime factor of the order of $\mathbb{F}_p^*$ is only 88 bits in length. We also note that an easier way to launch a large subgroup attack is to use the supersingular curve $E : y^2 = x^3 - 3x$ over $\mathbb{F}_p$ of order $p + 1 = 2^{521}$. For this curve, discrete logarithms can be computed extremely efficiently using the Pohlig-Hellman algorithm. However, this speedup is not applicable for all primes. For example, the supersingular curve $E : y^2 = x^3 - 3x$ over $\mathbb{F}_p$ where $p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$ is the 384-bit NIST prime has a subgroup of 188-bit prime order

$$r = 21345864075709046259206863397523075754412495433135288906 1,$$

and hence the large subgroup attack cannot be immediately applied. However, the embedding degree of E is two, and the discrete logarithm problem can be efficiently mapped via pairings to $\mathbb{F}_{p^2}^*$.

5.5 Concluding remarks

We have demonstrated that invalid-curve attacks can be extended to hyperelliptic curve cryptosystems that use genus-2 curves defined over prime fields and the addition formula $\mathcal{F}_{2a}$. The attacks can be extended in various ways. First, the attacks are applicable when genus-2 curves defined over binary fields are used together with the addition formulas given in [5, Section 14.3.2]. Furthermore, the attacks can be extended to genus-3 hyperelliptic curves using the addition formulas in [5, Section 14.6.1, Section 14.6.2]. More interestingly, it is also possible to use invalid singular hyperelliptic curves to mount attacks analogously as was done in Section 5.4.2 with singular elliptic curves. For example, suppose $\mathcal{H} : y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$ is a non-singular genus-2 hyperelliptic curve defined over $\mathbb{F}_q$. By the proof of Theorem 5.3.4, there exist at least $q-6$ invalid singular curves $\mathcal{IH}$ relative to $\mathcal{H}$ and $\mathcal{F}_{2a}$ such that $\mathcal{IH} : y^2 = (x+a)^2(x^3 + k_2x^2 + k_1x + k_0)$ and $x^3 + k_2x^2 + k_1x + k_0$ has no repeated roots; let $\mathcal{E} : y^2 = x^3 + k_2x^2 + k_1x + k_0$ be the corresponding elliptic curve. Let $J_{ns}(\mathbb{F}_q)$ denote the group of degree zero divisor classes of $\mathcal{IH}$ over $\mathbb{F}_q$ such that the support of the divisors does not contain the singular point $(-a, 0)$. An explicit isomorphism from $J_{ns}(\mathbb{F}_q)$ to the group $\mathcal{E}(\mathbb{F}_q)$, which is induced by $\rho : \mathcal{IH} \setminus \{(-a,0)\} \to \mathcal{E}$ such that

$\rho(x, y) = (x, y/(x+a))$ and $\rho(\infty) = \infty$, can be used to map the discrete logarithm problem in $J_{ns}(\mathbb{F}_q)$ to $\mathcal{E}(\mathbb{F}_q)$. Hence, if the validation check is omitted in a cryptographic protocol employing $\mathcal{H}$ that requires a party $\hat{B}$ to compute $b\bar{D}$ for some incoming $\bar{D} \in J_{\mathcal{H}}$, then an adversary $\mathcal{M}$ can recover the static private key b of $\hat{B}$ in time $\mathcal{O}(\sqrt{q})$ rather than $\mathcal{O}(q)$.

Altogether, these attacks emphasize the importance of validating public keys in discrete logarithm cryptosystems.

Chapter 6

A New Protocol for the Nearby Friend Problem

In this chapter, we investigate the so-called nearby friend problem. The problem has emerged in the context of location-based services such as social networking and is closely related to the issue of location privacy. In particular, we are interested in the question of how Alice can efficiently determine whether a friend Bob is at a nearby location or not. This has to be achieved without a third party and where Alice neither reveals any information about her own location nor can she extract any information about Bob's actual location when they are not nearby. Similarly, no eavesdropper should be able to gain any information about their actual locations, whether they are actually nearby or not. The problem becomes more challenging as both Alice and Bob are restricted in computational power and communication bandwidth. Starting from an earlier work by Zhong et al., we formalize the protocol definition and the security model and then propose a new protocol that solves the problem in the proposed security model. An interesting feature of the protocol is that it does not depend on any other cryptographic primitive, thus providing a new approach to solve the nearby friend problem. Our basic protocol and its extensions compare favorably with the earlier solutions for this problem. The protocol might be of use in other privacy-preserving applications.

The remainder of this chapter is organized as follows. In Section 6.1 we introduce a formal definition of the nearby friend problem and its security model. In Section 6.2 we propose the basic protocol $\mathcal{NFP}$-I and discuss its security. In Section 6.3 we discuss some extensions of the basic primitive and compare our protocols with the currently best known protocols. One of our extension techniques also applies to some of the previously known protocols and, in particular, improves the functionality and the performance of the Pierre protocol. Finally we conclude in Section 6.4 with some open problems in this emerging area.

The results of this chapter appeared in [18], a joint work with S. Chatterjee and A. Menezes.

6.1 Problem definition and security model

In this section we propose a formal definition and security model for the nearby friend problem. The protocol definition is developed from the informal description of the problem in [94].

6.1.1 Nearby friend problem

Let $\mathcal{L}$ be the publicly known set of all possible locations. The elements of $\mathcal{L}$ can be expressed in two-dimensional coordinates such as floating point numbers or integers to represent some geographical location or block on the surface of the earth. Alternatively, they can be some description of a location such as the ZIP code or the street address. We assume there exists a natural notion of nearbyness for the elements of $\mathcal{L}$ such that given $L_1, L_2 \in \mathcal{L}$ it is easy to decide whether they are nearby or not. For example, when locations are represented by Cartesian coordinates, given two such elements $L_1, L_2 \in \mathcal{L}$ one can compute the Euclidean distance between them. In this context, we say two location points $L_1, L_2 \in \mathcal{L}$ are nearby (denoted as $L_1 \equiv L_2$) if the Euclidean distance between L_1 and L_2 is less than or equal to some threshold distance d.

A scheme to solve the nearby friend problem is a two-party protocol $\mathcal{NFP}$ between an initiator $\mathcal{I}$ and a responder $\mathcal{R}$. It is assumed that both parties agree upon a description of $\mathcal{L}$ (and the corresponding notion of nearbyness) before running the protocol. It is also assumed that the parties establish a communication link which is authenticated, but not necessarily confidential, before executing the protocol. Such an authenticated link can be established, for example, by deriving a session key using a standard key agreement protocol and then MACing all messages using that session key. $\mathcal{I}$'s input to $\mathcal{NFP}$ is her secret location information $L_{\mathcal{I}} \in \mathcal{L}$ while $\mathcal{R}$'s input is his secret location information $L_{\mathcal{R}} \in \mathcal{L}$. Both parties exchange some messages based on their secret input and what they receive from the other party. At the end, the protocol generates an output for the initiator, namely the initiator will learn whether $L_{\mathcal{I}} \equiv L_{\mathcal{R}}$ or not. (The responder does not learn any information about the initiator's location $L_{\mathcal{I}}$.)

6.1.2 Security model

To model (all) possible leakage of information, we consider two different kinds of computationally-bounded adversaries – passive and active. In the former case, the target of an eavesdropper

who observes the exchanged messages in a protocol run is either to extract any information about $\mathcal{I}$ or $\mathcal{R}$'s true location or to just decide whether they are actually nearby or not. The case of a passive adversary is relevant in our context as we only assume the existence of an authenticated link between $\mathcal{I}$ and $\mathcal{R}$. The work of Zhong et al., in contrast, assumes the existence of a communication channel (such as TLS connection) which provides both confidentiality and authentication. To prevent active attacks, on the other hand, one of $\mathcal{I}$ or $\mathcal{R}$ is treated as the adversary where the goal is to extract any information about the other party's location. In the following description, a protocol transcript stands for the collection of all the messages transmitted between the two parties during a typical protocol run.

Passive Adversary.

We would like to provide the adversary with as much power as possible while limiting its task to the minimum. Intuitively the goal is to show that even if the adversary has some a priori information about $\mathcal{I}$'s (or $\mathcal{R}$'s) possible locations it should not be able to distinguish the protocol transcript for one location of $\mathcal{I}$ (or $\mathcal{R}$) from any other. We also want the assurance that after observing the protocol messages the adversary should not be able to infer whether the participants are actually nearby or not. In other words, the protocol transcript when $\mathcal{I}$ and $\mathcal{R}$ are actually nearby should not be distinguishable from a protocol transcript when they are not. This is formally modeled through the following two games between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. In the following and all the subsequent games between $\mathcal{A}$ and $\mathcal{C}$ it is assumed that the set of location points $\mathcal{L}$ and the measure of nearbyness are publicly known and hence available to $\mathcal{A}$.

The initiator's privacy. This is modelled by the following game.

$\mathcal{G}_1$*:* $\mathcal{A}$ chooses two distinct locations $L_{\mathcal{I},0}, L_{\mathcal{I},1} \in \mathcal{L}$ for $\mathcal{I}$ and one location $L_{\mathcal{R}} \in \mathcal{L}$ for $\mathcal{R}$.[1] The challenger $\mathcal{C}$ selects $b \in_R \{0,1\}$ and provides $\mathcal{A}$ with a protocol transcript corresponding to $L_{\mathcal{I},b}$ as $\mathcal{I}$'s location and $L_{\mathcal{R}}$ as $\mathcal{R}$'s location. $\mathcal{A}$ outputs a guess b' and wins if $b = b'$. We define the advantage of the adversary $\mathcal{A}$ in attacking the scheme $\mathcal{NFP}$ with respect to $\mathcal{I}$'s location as

$$\mathsf{Adv}^{\mathcal{I}}_{\mathcal{NFP},\mathcal{A}} = \left| \Pr[b = b'] - \frac{1}{2} \right|.$$

We say that $\mathcal{NFP}$ preserves location privacy for $\mathcal{I}$ against a passive adversary if $\mathsf{Adv}^{\mathcal{I}}_{\mathcal{NFP},\mathcal{A}}$ is negligible for all possible $\mathcal{A}$.

[1] It should be understood that there are no further restrictions on $\mathcal{A}$'s choices for $L_{\mathcal{I},0}, L_{\mathcal{I},1}$ and $L_{\mathcal{R}}$. For example, $\mathcal{A}$ can choose $L_{\mathcal{R}}$ to be nearby to $L_{\mathcal{I},0}$ or $L_{\mathcal{I},1}$ (or both).

The responder's privacy. The aim of the adversary in this case is similar to that in the former. Formally we have:

$\mathcal{G}_2$*:* $\mathcal{A}$ chooses one location $L_{\mathcal{I}} \in \mathcal{L}$ for $\mathcal{I}$ and two distinct locations $L_{\mathcal{R},0}, L_{\mathcal{R},1} \in \mathcal{L}$ for $\mathcal{R}$. $\mathcal{C}$ selects $b \in_R \{0,1\}$ and provides $\mathcal{A}$ with a protocol transcript corresponding to $L_{\mathcal{I}}$ as $\mathcal{I}$'s location and $L_{\mathcal{R},b}$ as $\mathcal{R}$'s location. $\mathcal{A}$ outputs a guess b' and wins if $b = b'$. We define the advantage of the adversary $\mathcal{A}$ in attacking the scheme $\mathcal{NFP}$ with respect to $\mathcal{R}$'s location as

$$\mathsf{Adv}^{\mathcal{R}}_{\mathcal{NFP},\mathcal{A}} = \left| \Pr[b = b'] - \frac{1}{2} \right|.$$

We say that $\mathcal{NFP}$ preserves location privacy for $\mathcal{R}$ against a passive adversary if $\mathsf{Adv}^{\mathcal{R}}_{\mathcal{NFP},\mathcal{A}}$ is negligible for all possible $\mathcal{A}$.

A protocol secure against a passive adversary in terms of the above two games gives the assurance that after observing a protocol transcript neither can the adversary distinguish between two locations of $\mathcal{I}$ (or $\mathcal{R}$) nor would it be possible to decide whether the participants are actually nearby or not.

Active Adversary.

The aim here is to model the case of malicious participants. In particular, we are interested in formulating the security assurance of $\mathcal{I}$ vis-a-vis $\mathcal{R}$ and vice versa. Since one of the participants in the protocol is now treated as the adversary, (s)he is allowed to behave arbitrarily in the protocol run.

Initiator's privacy wrt the responder. This is stronger than the case of $\mathcal{G}_1$. Namely, we have the following:

$\mathcal{G}_3$*:* $\mathcal{A}$ plays the role of $\mathcal{R}$ and chooses two distinct locations $L_0, L_1 \in \mathcal{L}$ for $\mathcal{I}$. $\mathcal{C}$ picks L_b where $b \in_R \{0,1\}$ as $\mathcal{I}$'s location point and executes the protocol with $\mathcal{A}$. $\mathcal{A}$'s task is to output a guess b' of b and it wins if $b = b'$. The advantage of the adversary $\mathcal{A}$ in the role of the responder in attacking the scheme $\mathcal{NFP}$ with respect to $\mathcal{I}$'s location is defined as

$$\mathsf{Adv}^{\mathcal{I}}_{\mathcal{NFP},\mathcal{R}} = \left| \Pr[b = b'] - \frac{1}{2} \right|.$$

We say that $\mathcal{NFP}$ preserves location privacy for $\mathcal{I}$ against an active adversary if $\mathsf{Adv}^{\mathcal{I}}_{\mathcal{NFP},\mathcal{R}}$ is negligible for all possible $\mathcal{A}$.

Responder's privacy wrt the initiator. Note that it is easy for a malicious $\mathcal{I}$ to verify a guess $L'_{\mathcal{R}}$ of $\mathcal{R}$'s location by choosing her own location as nearby to $L'_{\mathcal{R}}$. This kind of *probing attack* cannot be prevented because of the very nature of the problem. The only safeguard for $\mathcal{R}$ is not to participate in the protocol if he has some reason to believe that $\mathcal{I}$ is behaving suspiciously. The aim here is rather to ensure that a malicious $\mathcal{I}$ does not learn any information (other than what the protocol entitles her) about $\mathcal{R}$'s location $L_{\mathcal{R}}$ when the latter is uniformly distributed in $\mathcal{L}$. This is captured in the following game.

$\mathcal{G}_4$*:* $\mathcal{A}$ initiates the protocol and provides the challenger with a message as in a valid protocol run. In response $\mathcal{C}$ chooses a location point $L_{\mathcal{R}}$ for $\mathcal{R}$ uniformly at random from the set of all possible locations $\mathcal{L}$ and completes the protocol with $\mathcal{A}$ based on that location. $\mathcal{A}$'s task is to output a guess $L'_{\mathcal{R}}$ of $\mathcal{R}$'s location and it wins if $L'_{\mathcal{R}} \equiv L_{\mathcal{R}}$. Suppose there are (at most) k locations in $\mathcal{L}$ which are considered as nearby to a particular location. The advantage of the adversary $\mathcal{A}$ in the role of the initiator in attacking the scheme $\mathcal{NFP}$ with respect to $\mathcal{R}$'s location is defined as

$$\mathsf{Adv}^{\mathcal{R}}_{\mathcal{NFP},\mathcal{I}} = \left| \Pr[L'_{\mathcal{R}} \equiv L_{\mathcal{R}}] - \frac{k+1}{|\mathcal{L}|} \right|.$$

For a justification of the threshold probability $(k+1)/|\mathcal{L}|$, see the argument for Claim 3 in Section 6.2. We say that $\mathcal{NFP}$ preserves location privacy for $\mathcal{R}$ against an active adversary if $\mathsf{Adv}^{\mathcal{R}}_{\mathcal{NFP},\mathcal{I}}$ is negligible for all possible $\mathcal{A}$.

6.2 The $\mathcal{NFP}$-I protocol

We introduce our protocol to solve the nearby friend problem. The protocol, which we call $\mathcal{NFP}$-I, can also be viewed as a basic primitive to solve more complex private matching problems as we discuss later. The protocol resembles a Diffie-Hellman type of key exchange [24], and enjoys some attractive features in terms of performance and security.

6.2.1 Construction

A location corresponds to some point on a surface area such as the surface of the earth. In the protocol we assume that the set of locations consists of some distinct areas which are disjoint. Each such area defines a cell and each cell is identified by a unique description. This cell identifier can be the name of a city or the ZIP code of an area or some other description such as the name of an institution. For example, "Waterloo-University of Waterloo" or "Waterloo-Wilfrid Laurier University" can serve as the identifiers of two different cells. Such a description should be fixed a priori and two parties are said to be

PUBLIC INFORMATION: A set of locations $\mathcal{L}$, a multiplicative group $G = \langle g \rangle$ of prime order p, and an injective function $f : \mathcal{L} \to G^*$.
INPUT: The initiator $\mathcal{I}$'s input is her secret location $L_{\mathcal{I}}$ and the responder $\mathcal{R}$'s input is his secret location $L_{\mathcal{R}}$.

1. $\mathcal{I}$ computes $g_{\mathcal{I}} = f(L_{\mathcal{I}})$, chooses $\alpha \in_R \mathbb{Z}_p^*$, and computes $a = g_{\mathcal{I}}^{\alpha}$. $\mathcal{I}$ sends a to $\mathcal{R}$.
2. Upon receiving a from $\mathcal{I}$, $\mathcal{R}$ first computes $g_{\mathcal{R}} = f(L_{\mathcal{R}})$, chooses $\beta \in_R \mathbb{Z}_p^*$, and computes $b_1 = g_{\mathcal{R}}^{\beta}$ and $b_2 = a^{\beta}$. $\mathcal{R}$ sends (b_1, b_2) to $\mathcal{I}$.
3. Upon receiving (b_1, b_2) from $\mathcal{R}$, $\mathcal{I}$ computes $b_2' = b_1^{\alpha}$. $\mathcal{I}$ outputs nearby if $b_2' = b_2$; otherwise it outputs notnearby.

Figure 6.1: The $\mathcal{NFP}$-I protocol.

nearby if and only if they belong to the same cell. Zhong has proposed a similar strategy for the Wilfrid protocol [93].

$\mathcal{NFP}$-I protocol. In the protocol description, $\mathcal{L}$ is the set of locations, G is a multiplicative group of prime order p, and g is a publicly known generator of G. More concretely, we will take G to be the group $E(\mathbb{F}_q)$ of points on a randomly selected elliptic curve E of prime order p defined over a finite field of prime order $q \approx 2^{256}$. For example, E could be the curve P-256 specified in NIST's FIPS 186-3 standard [31]. Furthermore, we will assume that $3 \leq |\mathcal{L}| \leq 2^{40}$, and that the adversary's running time is bounded by 2^{80}. Under these conditions, the adversary is unable to compute discrete logarithms in G. Moreover, since the discrete logarithm problem in G is random self-reducible, the adversary is also unable to solve a single instance given up to 2^{40} instances of the problem.

Before executing the protocol, the parties need to establish an authenticated channel between them. We also need an injective function f that, given the description of a cell $L \in \mathcal{L}$, maps it to a unique non-identity element of G. Such a map-to-point function can be obtained in a way as suggested in [11, 12, 43] in the case where G is an elliptic curve group. For such an instantiation, f behaves like a random function in the sense that the discrete logarithm of $f(L)$ to the base g is presumably hard to compute; f is treated as a random oracle in the security argument of [11, 12]. We assume that f is public. The protocol is presented in Figure 6.1.

The protocol $\mathcal{NFP}$-I is quite efficient in terms of computation and communication bandwidth. After the execution of the protocol $\mathcal{I}$ will find $\mathcal{R}$ to be nearby if they belong to the same cell.

6.2.2 Security

We show that the protocol $\mathcal{NFP}$-I maintains all the security attributes defined in Section 6.1. In particular, $\mathcal{I}$'s location privacy is preserved information theoretically while that of $\mathcal{R}$ is based on the hardness of the decision Diffie-Hellman problem for a passive adversary and a variant of it in case of an active adversary.

Claim 1. The initiator's privacy is information theoretically secure with respect to a passive as well as an active adversary. That is, no adversary can win the game $\mathcal{G}_1$ or $\mathcal{G}_3$ with nonzero advantage even when it has unbounded computational power.

Argument. We establish the claim wrt $\mathcal{G}_1$; the case for $\mathcal{G}_3$ is analogous. $\mathcal{A}$ picks two distinct locations $L_{\mathcal{I}_0}, L_{\mathcal{I}_1}$ for $\mathcal{I}$ and one location $L_{\mathcal{R}}$ for $\mathcal{R}$. Suppose that $f(L_{\mathcal{I}_0}) = g_{\mathcal{I}_0}, f(L_{\mathcal{I}_1}) = g_{\mathcal{I}_1}$, and $f(L_{\mathcal{R}}) = g_{\mathcal{R}}$. Suppose that $\mathcal{C}$ selects $b = 0$; the case $b = 1$ is similar. Then $\mathcal{C}$ provides $\mathcal{A}$ with a transcript of the form $\langle g_{\mathcal{I}_0}^{\alpha}, g_{\mathcal{R}}^{\beta}, g_{\mathcal{I}_0}^{\alpha\beta}\rangle$ where α and β are randomly selected from $\mathbb{Z}_p^*$. But G is a cyclic group and hence $g_{\mathcal{I}_0} = g_{\mathcal{I}_1}^{k}$ for some $k \in \mathbb{Z}_p^*$. So the transcript can also be viewed as $\langle g_{\mathcal{I}_1}^{k\alpha}, g_{\mathcal{R}}^{\beta}, g_{\mathcal{I}_1}^{k\alpha\beta}\rangle$. Hence $\mathcal{A}$ cannot determine whether this transcript corresponds to $(L_{\mathcal{I}_0}, L_{\mathcal{R}})$ where the randomizer for $\mathcal{I}$ is α or to $(L_{\mathcal{I}_1}, L_{\mathcal{R}})$ where the randomizer for $\mathcal{I}$ is $k\alpha$. □

Recall that the decision Diffie-Hellman (DDH) problem in G is to decide, given $\langle g, g^{\alpha}, g^{\beta}, h\rangle$, whether h is equal to $g^{\alpha\beta}$, where $\alpha, \beta \in_R \mathbb{Z}_p$, and h is either $g^{\alpha\beta}$ or a random element of G. The decision Diffie-Hellman assumption in G asserts that the DDH problem is hard in G.

Claim 2. The location privacy of $\mathcal{R}$ is preserved with respect to a passive adversary under the decision Diffie-Hellman assumption in G where f is treated as a random oracle.

Reductionist argument. Given an adversary $\mathcal{A}$ with a non-negligible advantage ϵ in game $\mathcal{G}_2$ we show how to construct a DDH solver. The challenger is provided with a DDH problem instance $\langle g, g^{\alpha}, g^{\beta}, h\rangle$. $\mathcal{A}$ selects one location $L_{\mathcal{I}}$ for $\mathcal{I}$ and two distinct locations $L_{\mathcal{R}_0}$ and $L_{\mathcal{R}_1}$ for $\mathcal{R}$. At any point during the game $\mathcal{A}$ can ask for an evaluation of f on any location in $\mathcal{L}$. For each such distinct query $\mathcal{C}$ chooses a distinct random $x \in \mathbb{Z}_p^*$ and returns g^x as the output of f. Let's suppose $f(L_{\mathcal{I}}) = g^{x_{\mathcal{I}}}$, $f(L_{\mathcal{R}_0}) = g^{y_0}$ and $f(L_{\mathcal{R}_1}) = g^{y_1}$. $\mathcal{C}$ chooses a random $b \in \{0, 1\}$ and returns the transcript $\langle (g^{\alpha})^{x_{\mathcal{I}}}, (g^{\beta})^{y_b}, h^{x_{\mathcal{I}}}\rangle$. If $h = g^{\alpha\beta}$ then this is a proper protocol transcript for $L_{\mathcal{I}}, L_{\mathcal{R}_b}$. Otherwise, the transcript is independent of b. Thus, $\mathcal{A}$'s probability of success in the former case will be $1/2 \pm \epsilon$ and that in the latter case will be $1/2$. Hence, $\mathcal{C}$ has an advantage $\epsilon/2$ in solving the DDH problem. On the other hand, if DDH problem is easy in G then one can trivially break the protocol. Hence the security of $\mathcal{NFP}$-I in terms of $\mathcal{G}_2$ is equivalent to the hardness of DDH problem in G. □

Claim 3. No computationally bounded malicious initiator will be able to find the location of the responder with a probability of success greater than $2/|\mathcal{L}|$.

Heuristic argument. We first consider a probing attack. $\mathcal{A}$ guesses some location $L_1 \in \mathcal{L}$, computes $f(L_1) \in G^*$, and sets $\tilde{g} = f(L_1)^\alpha$ for some $\alpha \in \mathbb{Z}_p^*$ as its message. As a response the challenger chooses a location L_B uniformly at random from $\mathcal{L}$, computes $h = f(L_B) \in G^*$, chooses $\beta \in_R \mathbb{Z}_p^*$, and provides $(b_1, b_2) = (h^\beta, \tilde{g}^\beta)$ as the message coming from the responder. If $b_1^\alpha = b_2$ $\mathcal{A}$ returns L_1, otherwise it returns $L_2 \in_R \mathcal{L} \setminus \{L_1\}$. The probability of success for $\mathcal{A}$ in this game is

$$\frac{1}{|\mathcal{L}|} + \left(1 - \frac{1}{|\mathcal{L}|}\right)\left(\frac{1}{|\mathcal{L}| - 1}\right) = \frac{2}{|\mathcal{L}|}.$$

Since we cannot prevent this kind of probing attack we want the assurance that this is the best $\mathcal{A}$ can do.

Let $S = \{h_1, h_2, \ldots, h_n\}$ be the (random) subset of G which corresponds to the range of f. $\mathcal{A}$ knows S because $\mathcal{A}$ has access to f. Note that for any $\tilde{g} \in G^*$ and $i \in \{1, \ldots, n\}$ we have $h_i = \tilde{g}^{k_i}$ for some $k_i \in \mathbb{Z}_p^*$. But the randomness property of f assures that no computationally bounded $\mathcal{A}$ can find two distinct elements $h_i, h_j \in S$ so that $\mathcal{A}$ knows the discrete log of both h_i, h_j to the base $\tilde{g} \in G^*$ of its choice. If $\mathcal{A}$ can successfully find L_B given $(h^\beta, \tilde{g}^\beta)$ then that amounts to finding $h = f(L_B)$ where $L_B \in_R \mathcal{L}$ and where $\tilde{g}$ is chosen by $\mathcal{A}$. Let $h = \tilde{g}^\gamma$ for some $\gamma \in \mathbb{Z}_p^*$. We consider two mutually exclusive cases: (i) $\mathcal{A}$ knows γ and (ii) $\mathcal{A}$ does not know γ.

(i) $\mathcal{A}$ knows γ. In this case $\mathcal{A}$ can easily obtain h as $h = \tilde{g}^\gamma$. But $h = f(L_B)$ is chosen from S after $\mathcal{A}$ has output $\tilde{g}$. So the best $\mathcal{A}$ can do is to use a plug-and-pray strategy – choose a random $h' \in S$ and then set $\tilde{g} = (h')^k$ for some known k. The probability that $h = h'$ in this case is $1/|\mathcal{L}|$ and this corresponds to $L_1 = L_B$ in the probing attack discussed above.

(ii) $\mathcal{A}$ does not know γ. If $\mathcal{A}$ can successfully predict L_B then that amounts to finding $h \in S$ given $\langle \tilde{g}^{\gamma\beta}, \tilde{g}^\beta \rangle$ where $\tilde{g} \in G$ is first chosen by $\mathcal{A}$ after which $\beta \in_R \mathbb{Z}_p^*$ and $h = \tilde{g}^\gamma \in_R S$ are chosen by the responder. This is a hard problem when $\mathcal{A}$ has no role in the choice of $\tilde{g}$ and $h = \tilde{g}^\gamma$ comes from the whole group G. (This is the so-called divisible computational Diffie-Hellman (DCDH) problem where the task is to find $g^{x/y}$ given $\langle g, g^x, g^y \rangle$ for randomly selected $x, y \in \mathbb{Z}_p^*$. In [29] it has been shown that the DCDH problem is equivalent to the computational Diffie-Hellman problem.) We assume that DCDH remains hard even when the adversary is allowed to choose the base $\tilde{g}$ and $\tilde{g}^\gamma$ comes from the random subset S. So it would appear that the best strategy for $\mathcal{A}$ is to return a random guess $h' \in S$ of h. The probability of success in this case is again $1/|\mathcal{L}|$ and this corresponds to the case $L_2 = L_B$ in the probing attack. □

This heuristic argument can be further formalized in terms of a reduction to a *non-standard* version of the decision Diffie-Hellman problem as shown below. Abdalla and

Pointcheval introduced several such interactive assumptions in [1]. These are termed by the authors as chosen-basis decisional Diffie-Hellman assumptions (CDDH1 and CDDH2) and password-based chosen-basis decisional Diffie-Hellman assumptions (PCDDH1 and PCDDH2). They showed how one can use a PCDDH1 solver (resp. PCDDH2 solver) to solve the CDDH1 problem (resp. CDDH2 problem). They also provided lower bounds for CDDH1 and CDDH2 problem in the generic group model of Shoup [80]. However, Szydlo [85] observed that one can easily break the CDDH1 and CDDH2 problems. No such attack is yet known for the other two assumptions, namely PCDDH1 and PCDDH2.

Here we show how an adversary in terms of the game $\mathcal{G}_4$ against $\mathcal{NFP}$-I can be used to construct a PCDDH2 solver. Note that the reduction only goes in one way and it is not known whether an efficient adversary against PCDDH2 will also make $\mathcal{NFP}$-I vulnerable.

Password-based chosen-basis decisional Diffie-Hellman assumption (PCDDH2). Let $G = \langle g \rangle$ be a cyclic group of some prime order p, and let $\mathcal{P}$ be a random function from $\{1, \ldots, n\}$ into G. The problem is defined in terms of the following interactive game between an adversary $\mathcal{B}$ and a challenger $\mathcal{C}$.

$\mathcal{B}$, given oracle access to $\mathcal{P}$, chooses two elements $g_1, g_2 \in G$ and gives them to $\mathcal{C}$. $\mathcal{C}$ chooses a random $k \in \{1, \ldots, n\}$, a random $r \in \mathbb{Z}_p^*$, and a random $b \in \{0, 1\}$. $\mathcal{C}$ then obtains $u = \mathcal{P}(k)$ and computes $(g_1/u)^r$. If $b = 0$, it outputs $\langle g_2^r, (g_1/u)^r, k \rangle$, otherwise it outputs $\langle g_2^r, h, k \rangle$, where h is a random element of G. The task of $\mathcal{B}$ is to guess the value of b. The PCDDH2 assumption asserts that this problem is hard in G.

Reductionist argument. Given an adversary $\mathcal{A}$ with a non-negligible advantage against $\mathcal{NFP}$-I in $\mathcal{G}_4$, we show how one can construct an algorithm $\mathcal{B}$ to solve the PCDDH2 problem. $\mathcal{B}$ sets $|\mathcal{L}| = n$ and then uses $\mathcal{P}$ to simulate f. When $\mathcal{A}$ sends some $\tilde{g}$ as a message from the initiator it sets $g_1 = \mathbf{1}$, where $\mathbf{1}$ is the identity element of G and $g_2 = \tilde{g}$ and sends them to $\mathcal{C}$. When it receives the challenge $\langle g_3, g_4, k \rangle$ from $\mathcal{C}$, $\mathcal{B}$ sends $\langle g_4^{-1}, g_3 \rangle$ as the corresponding message in $\mathcal{NFP}$-I. If $g_4 = (g_1/u)^r$ then this corresponds to $\langle u^r, \tilde{g}^r \rangle$, which is a valid protocol transcript for a location L_B such that $f(L_B) = \mathcal{P}(k)$. Otherwise the transcript is independent of L_B. Hence, any advantage of $\mathcal{A}$ in finding L_B can be converted to an advantage of $\mathcal{B}$ to solve the PCDDH2 problem. $\square$

6.3 Extensions and comparisons

In this section we consider several extensions of the basic primitive. So far we have treated locations as detached cells and two parties are considered *nearby* if and only if they are in the same cell. This is perfectly acceptable for some applications where cells are few and far between (say the large cities of the world). In such a scenario if the protocol informs

$\mathcal{I}$ that $\mathcal{R}$ is *not nearby* then with overwhelming probability they are not in geographically close locations. However, this might not be the case if cells are dense. As an illustrative example consider the following situation. $\mathcal{I}$'s location identifier is "Waterloo-University of Waterloo-The Mathematics and Computer Building" and that of $\mathcal{R}$ is "Waterloo-University of Waterloo-Davis Centre". Since these two buildings are represented as different cells, $\mathcal{NFP}$-I will return them as *not nearby* even though they are physically quite close.

One way to model such scenarios is to allow a hierarchy of cells, where cells in the lowest level represent smallest areas (say a building), which are contained in the next level cells (say a street), and so on. Such a granular description was first proposed by Zhong in [93]. Returning to our previous example, using a two-level description $\mathcal{I}$'s location will be {"Waterloo-University of Waterloo", "Waterloo-University of Waterloo-The Mathematics and Computer Building"} and that of $\mathcal{R}$ {"Waterloo-University of Waterloo", "Waterloo-University of Waterloo-Davis Centre"}. $\mathcal{I}$ and $\mathcal{R}$ may first run the protocol on the higher-level set and if they are in the same cell then on the lower level. However, they can easily send all the information in a single run.

We have described the situation with $\ell = 2$ level hierarchy. However it is easy to extend it to any $\ell > 2$ depending upon the application. The modified version maintains all the security attributes of the original protocol because both $\mathcal{I}$ and $\mathcal{R}$ use different randomizers for different levels. The protocol will also maintain its better performance over Wilfrid.

For completeness we now describe the Wilfrid protocol and compare it with the $\mathcal{NFP}$-I protocol.

6.3.1 The Wilfrid protocol

The Wilfrid protocol adapts Freedman, Nissim and Pinkas' protocol (FNP04) [32] to solve the two-party set intersection problem. In such a scheme, there are two parties $\mathcal{I}$ and $\mathcal{R}$ with their private sets $\mathcal{L}_\mathcal{I} = \{L_{\mathcal{I}_1}, L_{\mathcal{I}_2}, \ldots, L_{\mathcal{I}_m}\}$ and $\mathcal{L}_\mathcal{R} = \{L_{\mathcal{R}_1}, L_{\mathcal{R}_2}, \ldots, L_{\mathcal{R}_n}\}$, respectively, where $\mathcal{L}_\mathcal{I}, \mathcal{L}_\mathcal{R} \subset \mathcal{L}$ for some domain $\mathcal{L}$. $\mathcal{I}$ and $\mathcal{R}$ exchange messages and at the end $\mathcal{I}$ learns $\mathcal{L}_\mathcal{I} \cap \mathcal{L}_\mathcal{R}$. Of course, the privacy of the users' sets should be preserved as much as possible. FNP04 relies on homomorphic encryption; if $\mathcal{E}$ is an encryption function with some domain $\mathcal{L}$ in which an addition operation is defined, then $\mathcal{E}$ is called *(additively) homomorphic* if given $\mathcal{E}(L_1)$ and $\mathcal{E}(L_2)$ one can efficiently compute $\mathcal{E}(L_1 + L_2)$. By convention, we will write $\mathcal{E}(L_1 + L_2) = \mathcal{E}(L_1)\mathcal{E}(L_2)$.

A high-level description of FNP04 to solve the set intersection problem is as follows. For simplicity, we assume that there is an injective embedding of $\mathcal{L}$ into $\mathbb{Z}_p$: $L_i \hookrightarrow z_i$, and we write without loss of generality that $\mathcal{L}_\mathcal{I} = \{z_{\mathcal{I}_1}, z_{\mathcal{I}_2}, \ldots, z_{\mathcal{I}_m}\}$ and $\mathcal{L}_\mathcal{R} = \{z_{\mathcal{R}_1}, z_{\mathcal{R}_2}, \ldots, z_{\mathcal{R}_n}\}$. Moreover, we assume that $\mathrm{H} : \mathbb{Z}_p \to G$ is an efficiently computable cryptographic hash

function. Initiator $\mathcal{I}$ uses a homomorphic encryption function $\mathcal{E}_A$ with domain $\mathbb{Z}_p$, and with respect to his public key A. $\mathcal{I}$ computes the polynomial

$$P(X) = \prod_{j=1}^{m}(X - z_{\mathcal{I}_j}) = \sum_{j=0}^{m} I_j X^{m-j} \in \mathbb{Z}_p[X],$$

and sends $\mathcal{E}_A(I_j)$ for $j = 0, \ldots, m$ to $\mathcal{R}$. $\mathcal{R}$ chooses a random number $r_k \in [1, p]$ and computes $C_k = \mathcal{E}_A(r_k P(z_{\mathcal{R}_k}) + z_{\mathcal{R}_k})$ for $k = 1, \ldots, n$, and sends C_k to $\mathcal{I}$. Note that C_k can be computed using the homomorphic properties of $\mathcal{E}_A$. In particular,

$$\begin{aligned} \mathcal{E}_A(P(z_{\mathcal{R}_k})) &= \prod_{j=0}^{m} \mathcal{E}_A(I_j)^{z_{\mathcal{R}_k}^{m-j}} \\ &= (((\mathcal{E}(I_0)^{z_{\mathcal{R}_k}} \mathcal{E}(I_1))^{z_{\mathcal{R}_k}} \cdots)^{z_{\mathcal{R}_k}} \mathcal{E}(I_{m-1}))^{z_{\mathcal{R}_k}} \mathcal{E}(I_m), \end{aligned}$$

and

$$C_k = \mathcal{E}_A(r_k P(z_{\mathcal{R}_k}) + z_{\mathcal{R}_k}) = \mathcal{E}_A(P(z_{\mathcal{R}_k}))^{r_k} \mathcal{E}_A(z_{\mathcal{R}_k}).$$

$\mathcal{I}$ decrypts C_k to $D_k = \mathcal{D}_A(C_k)$ and checks if $D_k \in \mathcal{L}_I$, which is expected to be the case if and only if $z_{\mathcal{R}_k} \in \mathcal{L}_{\mathcal{I}}$.

Now, we let G be a group of order p with generator g and describe a (randomized) homomorphic encryption scheme (CGS97) proposed by Cramer, Gennaro and Schoenmakers [21]. The encryption function is

$$\begin{aligned} \mathcal{E}_A : \mathbb{Z}_p &\to G \times G \\ m &\mapsto (g^r, A^{r+m}), \end{aligned}$$

where $A = g^a$ is the public key of some user and $r \in [1, p]$ is a randomly chosen integer. The corresponding decryption function is

$$\begin{aligned} \mathcal{D}_A : G \times G &\to \mathbb{Z}_p \\ (c_1, c_2) &\mapsto \mathrm{DLOG}_A(c_1^{-a} c_2), \end{aligned}$$

where $\mathrm{DLOG}_A(x)$ is the discrete logarithm of x in G with respect to the base A. One can check that

$$\mathcal{E}_A(m_1 + m_2) = \mathcal{E}_A(m_1)\mathcal{E}_A(m_2).$$

FNP04 can be modified so that $\mathcal{I}$ does not need to perform decryption in the last step of the protocol; see [32]. For example, if CGS97 is deployed in the protocol, then after computing $\mathcal{E}_A(P(z_{\mathcal{R}_k})) = (e_1, e_2)$, the responder $\mathcal{R}$ computes $C_k = (c_1, c_2) = (e_1, e_2 \cdot$

$\mathrm{H}(z_{\mathcal{R}_k})$), and sends it to $\mathcal{I}$. $\mathcal{I}$ now checks if $c_1^{-a}c_2 = \mathrm{H}(z)$ for some $z \in \mathcal{L}_\mathcal{I}$. Note that if $z_{\mathcal{R}_k} \in \mathcal{L}_\mathcal{I}$ then $P(z_{\mathcal{R}_k}) = 0$ and $(c_1, c_2) = (g^r, A^r \cdot \mathrm{H}(z_{\mathcal{R}_k}))$ for some $r \in [1, p]$. That is, if $z_{\mathcal{R}_k} \in \mathcal{L}_\mathcal{I}$ then $c_1^{-a}c_2 = \mathrm{H}(z_{\mathcal{R}_k})$, and if $z_{\mathcal{R}_k} \notin \mathcal{L}_\mathcal{I}$ then $c_1^{-a}c_2$ looks like a random element in G.

It follows that using CGS97 in the two-party set intersection protocol, $\mathcal{I}$ is required to perform $2(m+1)$ exponentiations in the first step and send $2(m+1)$ group elements. $\mathcal{R}$ is required to perform $2mn + 2n$ exponentiations and send $2n$ group elements. To determine the intersection $\mathcal{L}_\mathcal{I} \cap \mathcal{L}_\mathcal{R}$, $\mathcal{I}$ has to perform n exponentiations in the last step. Note that when the size of $\mathcal{L}$ is significantly smaller than p then one can further optimize the protocol by using the hash values of the z_i's, which would reduce the number of exponentiations performed by $\mathcal{R}$ to $2n + 2mn(\log_2 |\mathcal{L}|/\log_2 |G|)$.

The Wilfrid protocol now follows in a straightforward manner. We might assume that $\mathcal{I}$ and $\mathcal{R}$ in the Wilfrid protocol both use the level ℓ hierarchy of cells, i.e., $\mathcal{L}_\mathcal{I} = \mathcal{L}_\mathcal{R} = \ell$. For a 128-bit security level, the system parameters might be chosen such that $|\mathcal{L}| = 2^{40}$ and $|G| = 2^{256}$. It follows from our argument above that if CGS97 is deployed in the Wilfrid protocol, then $\mathcal{I}$ is required to perform $2(\ell+1)$ exponentiations in the first step and send $2(\ell+1)$ group elements. $\mathcal{R}$ is required to perform $2\ell + 5\ell^2/16$ exponentiations and send 2ℓ group elements. To decide whether $\mathcal{R}$ is nearby, $\mathcal{I}$ has to perform at most ℓ (and at least 1) exponentiation.

In Table 6.1 we compare our protocol $\mathcal{NFP}$-I with the Wilfrid protocol, the currently best protocol known that achieves the same functionality.

	Alice → Bob		Bob → Alice		decide whether nearby:
	# exps	message size	# exps	message size	# exps by Alice
Wilfrid	$2(\ell+1)$	$2(\ell+1)$	$2\ell + 5\ell^2/16$	2ℓ	$\leq \ell$
$\mathcal{NFP}$-I	ℓ	ℓ	2ℓ	2ℓ	$\leq \ell$

Table 6.1: Performance comparison between Wilfrid and $\mathcal{NFP}$-I when implemented in a group G where the discrete log problem is assumed to be hard. Both protocols require two communication steps. The message size is the number of elements of G. We assume that the hierarchy level is ℓ and that $|\mathcal{L}| = 2^{40}$ and $|G| = 2^{256}$.

6.3.2 The Pierre protocol

For still some other applications it might be more useful to consider a contiguous area. Suppose $\mathcal{I}$ is at some geographical location on the surface of the earth and is interested to know whether $\mathcal{R}$ is within a disc of radius d centered at her own location. In other words,

$\mathcal{I}$ should be able to find $\mathcal{R}$ as *nearby* as long as the Euclidean distance between the two does not exceed d.

The Lester protocol in [94] provides a solution to this problem. However, the solution is not fully satisfactory. To determine whether $\mathcal{R}$ is within a distance d, $\mathcal{I}$ needs to solve a discrete log problem in the range of $[0, d \cdot 2^t]$, where t is some safety factor chosen by $\mathcal{R}$. On the other side, $\mathcal{R}$ has no knowledge (and control) on d. In fact, the protocol allows $\mathcal{I}$ to extract some information about $\mathcal{R}$'s location even when the distance between them is greater than d. For example, with twice (resp. three times) as much work $\mathcal{I}$ can determine whether $\mathcal{R}$ is within a distance $2d$ (resp. $3d$).

In the Pierre protocol [94], instead of considering a disc around $\mathcal{I}$, a contiguous surface area is divided into square grid cells of side length d and the coordinates of each party are expressed in integral units of d. After execution of the protocol, $\mathcal{I}$ will find $\mathcal{R}$ if they are in the same cell or $\mathcal{R}$ is in one of the eight adjacent cells.

For completeness, we next describe the Pierre protocol.

Assume that locations are expressed in integer coordinates (X, Y), with the positive X and Y axes making an angle of 90 degrees, and that each square grid cell has side length d. $\mathcal{I}$ and $\mathcal{R}$ are considered nearby if either $\mathcal{R}$ is in the same cell as $\mathcal{I}$, or $\mathcal{R}$ is in one of the eight cells that are adjacent to $\mathcal{I}$'s cell. In the protocol, each cell is identified with its bottom left corner, and each location (X_A, Y_A) is identified with $(x_A, y_A) = (\lfloor X_A/d \rfloor, \lfloor Y_A/d \rfloor)$. In this representation, if $\mathcal{I}$'s and $\mathcal{R}$'s locations are identified with $\mathcal{L}_\mathcal{I} = (x_\mathcal{I}, y_\mathcal{I})$ and $\mathcal{L}_\mathcal{R} = (x_\mathcal{R}, y_\mathcal{R})$ then they are considered nearby if and only if the square of the Euclidean distance between $\mathcal{L}_\mathcal{I}$ and $\mathcal{L}_\mathcal{R}$ is one of $0, 1$ or 2, i.e., if and only if

$$D_{\mathcal{I},\mathcal{R}} = (x_\mathcal{I} - x_\mathcal{R})^2 + (y_\mathcal{I} - y_\mathcal{R})^2 \in \{0, 1, 2\}.$$

For example, if $\mathcal{I}$ is in the cell labelled I_0 as shown in Figure 6.2, then after executing the protocol $\mathcal{I}$ will find $\mathcal{R}$ to be nearby if and only if $\mathcal{R}$ is in one of the nine cells $\mathcal{I}_0$, $\mathcal{I}_1$, $\ldots$, $\mathcal{I}_8$.

The Pierre protocol relies on homomorphic encryption (see Section 6.3.1) and can be described as follows. Suppose $\mathcal{I}$ has public key A, and $\mathcal{I}$ and $\mathcal{R}$'s locations are identified with $\mathcal{L}_\mathcal{I} = (x_\mathcal{I}, y_\mathcal{I})$ and $\mathcal{L}_\mathcal{R} = (x_\mathcal{R}, y_\mathcal{R})$. $\mathcal{I}$ computes $\mathcal{E}_A(x_\mathcal{I}^2 + y_\mathcal{I}^2)$, $\mathcal{E}_A(2x_\mathcal{I})$, $\mathcal{E}_A(2y_\mathcal{I})$ and sends these three encrypted texts to $\mathcal{R}$. $\mathcal{R}$ chooses three random integers r_0, r_1, r_2 and, using homomorphic properties of $\mathcal{E}_a$, computes

$$\begin{aligned}
\mathcal{E}_A(D_{\mathcal{I},\mathcal{R}}) &= \frac{\mathcal{E}_A(x_\mathcal{I}^2 + y_\mathcal{I}^2)\mathcal{E}_A(x_\mathcal{R}^2 + y_\mathcal{R}^2)}{\mathcal{E}_A(2x_\mathcal{I})^{x_\mathcal{R}}\mathcal{E}_A(2y_\mathcal{I})^{y_\mathcal{R}}}, \\
\mathcal{E}_A(r_0 D_{\mathcal{I},\mathcal{R}}) &= \mathcal{E}_A(D_{\mathcal{I},\mathcal{R}})^{r_0}, \\
\mathcal{E}_A(r_1(D_{\mathcal{I},\mathcal{R}} - 1)) &= \left(\frac{\mathcal{E}_A(D_{\mathcal{I},\mathcal{R}})}{\mathcal{E}_A(1)}\right)^{r_1}, \\
\mathcal{E}_A(r_2(D_{\mathcal{I},\mathcal{R}} - 2)) &= \left(\frac{\mathcal{E}_A(D_{\mathcal{I},\mathcal{R}})}{\mathcal{E}_A(1)\mathcal{E}_A(1)}\right)^{r_2},
\end{aligned}$$

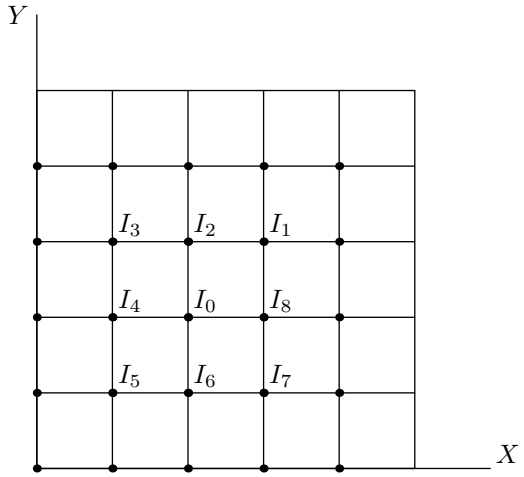


Figure 6.2: Discretization of the surface into square cells.

and sends $C_0 = \mathcal{E}_A(r_0 D_{\mathcal{I},\mathcal{R}}), C_1 = \mathcal{E}_A(r_1(D_{\mathcal{I},\mathcal{R}} - 1))$ and $C_2 = \mathcal{E}_A(r_2(D_{\mathcal{I},\mathcal{R}} - 2))$ to $\mathcal{I}$. $\mathcal{I}$ decides that they are nearby if and only if one of the C_i's corresponds to the encryption of 0.

Note that when CGS97 is deployed in the Pierre protocol, we can argue similarly as in Section 6.3.1 and conclude that $\mathcal{I}$ is required to perform 6 exponentiations and send 6 group elements to $\mathcal{R}$. $\mathcal{R}$ is required to perform $8 + 4(\log_2 |\mathcal{L}|/\log_2 |G|)$ exponentiations and send 6 group elements. To decide whether $\mathcal{R}$ is nearby, $\mathcal{I}$ has to perform at most 3 exponentiations.

6.3.3 $\mathcal{NFP}$-II: $\mathcal{NFP}$ with hexagonal discretization

In the following we use a slightly different discretization technique, suggested to us by David Wagner (University of Waterloo). Instead of using square grid cells we tile the surface by regular hexagons (see Figure 6.3). This gives a better approximation of a circle around $\mathcal{I}$'s location.

Assume that the locations are expressed in integer coordinates (X, Y), with the positive X and Y axes making an angle of 60 degrees and each hexagonal cell has side length d. With this representation, the centers of the cells will be at the points (dx, dy) with x, y integers and $x - y \equiv 0 \pmod 3$. Given a location (X_A, Y_A) we can find the center of the corresponding cell (x_A, y_A) as follows. Let $x_0 = \lfloor X_A/d \rfloor$ and $y_0 = \lfloor Y_A/d \rfloor$. If $x_0 - y_0 \equiv 1 \pmod 3$ then $(x_A, y_A) = (x_0, y_0+1)$; else if $x_0 - y_0 \equiv 2 \pmod 3$ then $(x_A, y_A) = (x_0+1, y_0)$; else if $X_A + Y_A \leq d(x_0 + y_0 + 1)$ then $(x_A, y_A) = (x_0, y_0)$; else $(x_A, y_A) = (x_0 + 1, y_0 + 1)$.

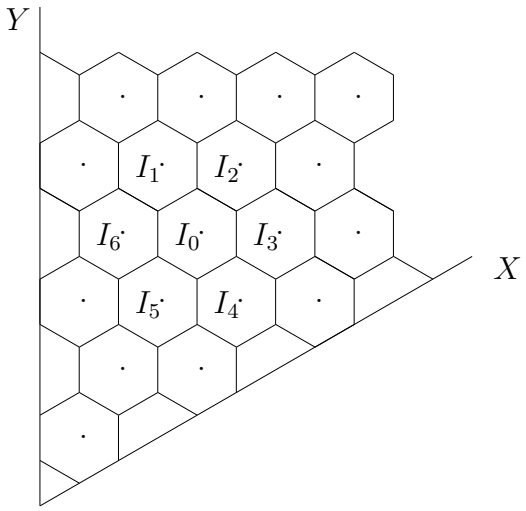


Figure 6.3: Discretization of the surface into hexagonal cells.

In the protocol, $\mathcal{I}$ first finds the center of the cell she is in (labelled I_0) and then the center of the six adjacent cells (labelled $I_1, \ldots, I_6$ as shown in Figure 6.3). After executing the protocol $\mathcal{I}$ will find $\mathcal{R}$ to be nearby if and only if he is in one of these seven cells. She will also learn exactly in which cell he is. Let $\mathcal{L}$ be the set of the hexagonal cells where each cell is identified by its center point. Note that $\mathcal{L}$ depends upon d and $\mathcal{I}$ and $\mathcal{R}$ must agree upon this value prior to running the protocol.

Let G be a multiplicative group of prime order p and let g be a publicly known generator of G. In the protocol description, $f : \mathcal{L} \rightarrow G^*$ is an injective function that takes as input a center point of a cell and outputs a random element in G^*. Such a map-to-point function can be constructed in an analogous way as described in Section 6.2.1. Before executing the protocol, the parties establish an authenticated channel between them. The protocol is presented in Figure 6.4.

Security. In the protocol $\mathcal{NFP}$-II, $\mathcal{I}$ uses a separate randomizer α_j for each I_j, hence the argument in Claim 1 for $\mathcal{NFP}$-I can be trivially extended to $\mathcal{NFP}$-II. So the location privacy of the initiator is information theoretically preserved with respect to a passive as well as an active adversary. Similarly, for a passive adversary we can extend the argument in Claim 2 to show that breaking the location privacy of $\mathcal{R}$ is equivalent to solving the DDH problem in G. Thus $\mathcal{NFP}$-II provides the same assurances as $\mathcal{NFP}$-I in terms of the security games $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$.

However, the situation is a little different when we consider the location privacy of the responder with respect to a malicious initiator. $\mathcal{I}$ can now deviate from the protocol and choose seven different cells (not necessarily adjacent) as her input. If $\mathcal{R}$ happens to be

PUBLIC INFORMATION: A description of the set of locations $\mathcal{L}$, a multiplicative group $G = \langle g \rangle$ of prime order p, and an injective function $f : \mathcal{L} \rightarrow G^*$.
INPUT: The initiator $\mathcal{I}$'s input is her secret cell and the six adjacent cells $L_{\mathcal{I}} = (I_0, I_1, I_2, I_3, I_4, I_5, I_6)$ and the responder $\mathcal{R}$'s input is his secret cell $L_{\mathcal{R}}$.

1. For $0 \leq j \leq 6$, $\mathcal{I}$ computes $g_{\mathcal{I},j} = f(I_j)$, chooses $\alpha_j \in_R \mathbb{Z}_p^*$, and computes $a_j = g_{\mathcal{I},j}^{\alpha_j}$. $\mathcal{I}$ sends $(a_0, a_1, a_2, a_3, a_4, a_5, a_6)$ to $\mathcal{R}$.

2. Upon receiving $(a_0, a_1, a_2, a_3, a_4, a_5, a_6)$ from $\mathcal{I}$, $\mathcal{R}$ first computes $g_{\mathcal{R}} = f(L_{\mathcal{R}})$, chooses $\beta \in_R \mathbb{Z}_p^*$, and then computes $b = g_{\mathcal{R}}^{\beta}$ and $h_j = a_j^{\beta}$ for $0 \leq j \leq 6$. $\mathcal{R}$ sends to $\mathcal{I}$ the tuple $(b, h_0, h_1, h_2, h_3, h_4, h_5, h_6)$.

3. Upon receiving $(b, h_0, h_1, h_2, h_3, h_4, h_5, h_6)$ from $\mathcal{R}$, $\mathcal{I}$ computes $h'_j = b^{\alpha_j}$ for $0 \leq j \leq 6$. $\mathcal{I}$ outputs nearby if $h_j = h'_j$ for some $0 \leq j \leq 6$. Otherwise it outputs notnearby.

Figure 6.4: The $\mathcal{NFP}$-II protocol.

in one of these cells then $\mathcal{I}$ can easily detect that in the verification step (i.e., Step 3) of $\mathcal{NFP}$-II. If not it returns a random cell from the rest. So its probability of success in the probing attack will be $8/|\mathcal{L}|$ (it is $2/|\mathcal{L}|$ in $\mathcal{NFP}$-I). In Claim 3 we argued that the probing attack is the best strategy for a malicious initiator in $\mathcal{NFP}$-I. A similar argument can be put forth for $\mathcal{NFP}$-II also based on related but stronger complexity assumptions.

Performance. In $\mathcal{NFP}$-II, $\mathcal{I}$ performs seven group exponentiations to compute her message while $\mathcal{R}$ performs eight exponentiations to generate his message. $\mathcal{I}$ needs at most seven more exponentiations to decide whether $\mathcal{R}$ is nearby or not. The message from $\mathcal{I}$ to $\mathcal{R}$ consists of seven elements of G and that from $\mathcal{R}$ to $\mathcal{I}$ consists of eight elements of G. After executing the protocol $\mathcal{I}$ will find $\mathcal{R}$ to be nearby if they belong to the same cell or if $\mathcal{R}$ is in one of the six cells adjacent to $\mathcal{I}$'s. $\mathcal{I}$ will also learn that $\mathcal{R}$ is in the jth cell wrt $\mathcal{I}$ if $h'_j = h_j$, $0 \leq j \leq 6$.

We note that the Pierre protocol can also be modified to use hexagonal tilling. In fact, using this technique, the Pierre protocol will gain a better performance in addition to having a better approximation of a circle around $\mathcal{I}$'s location. We provide more details in the next section.

6.3.4 The Pierre protocol with hexagonal discretization

Let $\mathcal{I}$ and $\mathcal{R}$ be at locations $\mathcal{L}_{\mathcal{I}} = (X_{\mathcal{I}}, Y_{\mathcal{I}})$ and $\mathcal{L}_{\mathcal{R}} = (X_{\mathcal{R}}, Y_{\mathcal{R}})$. Suppose that $\mathcal{L}_{\mathcal{I}}$ and $\mathcal{L}_{\mathcal{R}}$ are identified with $\mathcal{L}_{\mathcal{I}}^S = (x_{\mathcal{I}}^S, y_{\mathcal{I}}^S)$ and $\mathcal{L}_{\mathcal{R}}^S = (x_{\mathcal{R}}^S, y_{\mathcal{R}}^S)$ when using square cells; and with

$\mathcal{L}_{\mathcal{I}}^H = (x_{\mathcal{I}}, y_{\mathcal{I}})$ and $\mathcal{L}_{\mathcal{R}}^H = (x_{\mathcal{R}}, y_{\mathcal{R}})$ when using hexagonal cells.

Recall that when discretizing the plane using square cells $\mathcal{I}$ and $\mathcal{R}$ are nearby if and only if $D_{\mathcal{I},\mathcal{R}}^S \in \{0,1,2\}$ where $D_{\mathcal{I},\mathcal{R}}^S$ is the square of the Euclidean distance between $\mathcal{L}_{\mathcal{I}}^S$ and $\mathcal{L}_{\mathcal{R}}^S$. Therefore, $\mathcal{R}$ has to send 3 encrypted texts $\mathcal{E}_A(r_0 D_{\mathcal{I},\mathcal{R}}^S)$, $\mathcal{E}_A(r_1(D_{\mathcal{I},\mathcal{R}}^S - 1))$ and $\mathcal{E}_A(r_2(D_{\mathcal{I},\mathcal{R}}^S - 2))$. On the other hand, discretizing the plane using hexagonal cells we can show that $\mathcal{I}$ and $\mathcal{R}$ are nearby if and only if $D_{\mathcal{I},\mathcal{R}}^H \in \{0,3\}$ where $D_{\mathcal{I},\mathcal{R}}^S$ is the square of the Euclidean distance between $\mathcal{L}_{\mathcal{I}}^H$ and $\mathcal{L}_{\mathcal{R}}^H$. Therefore, it suffices for $\mathcal{R}$ to send only 2 (instead of 3) encrypted texts $\mathcal{E}_A(r_0 D_{\mathcal{I},\mathcal{R}}^H)$ and $\mathcal{E}_A(r_1(D_{\mathcal{I},\mathcal{R}}^H - 3))$. More precisely, $D_{\mathcal{I},\mathcal{R}}^H$ and $\mathcal{E}_A(D_{\mathcal{I},\mathcal{R}}^H)$ can be computed as follows:

$$\begin{aligned} D_{\mathcal{I},\mathcal{R}}^H &= ({x_{\mathcal{I}}}^2 + {y_{\mathcal{I}}}^2 + x_{\mathcal{I}} y_{\mathcal{I}}) - x_{\mathcal{I}}(2x_{\mathcal{R}} + y_{\mathcal{R}}) \\ &\quad -y_{\mathcal{I}}(2y_{\mathcal{R}} + x_{\mathcal{R}}) + ({x_{\mathcal{R}}}^2 + {y_{\mathcal{R}}}^2 + x_{\mathcal{R}} y_{\mathcal{R}}), \end{aligned}$$

and

$$\mathcal{E}_A({D^H}_{\mathcal{I},\mathcal{R}}) = \frac{\mathcal{E}_A({x_{\mathcal{I}}}^2 + {y_{\mathcal{I}}}^2 + x_{\mathcal{I}} y_{\mathcal{I}})\mathcal{E}_A({x_{\mathcal{R}}}^2 + {y_{\mathcal{R}}}^2 + x_{\mathcal{R}} y_{\mathcal{R}})}{\mathcal{E}_A(x_{\mathcal{I}})^{2x_{\mathcal{R}} + y_{\mathcal{R}}} \mathcal{E}_A(y_{\mathcal{I}})^{2y_{\mathcal{R}} + x_{\mathcal{R}}}}$$

In particular, in Figure 6.3 we have $I_0 = (4,4), I_1 = (3,6), I_2 = (5,5), I_3 = (6,3), I_4 = (5,2), I_5 = (3,3), I_6 = (2,5)$, and $D_{I_0,I_0}^H = 0$ and $D_{I_0,I_j}^H = 3$ for $j = 1,2,\ldots,6$.

Hence, if the Pierre protocol is modified to be used with hexagonal tilling (and CGS97 is deployed), then $\mathcal{R}$ is now required to perform $6 + 4(\log_2 |\mathcal{L}| / \log_2 |G|)$ exponentiations and send 4 group elements. To decide whether $\mathcal{R}$ is nearby, $\mathcal{I}$ has to perform at most 2 exponentiations; see Table 6.2.

6.3.5 Comparisons

In Table 6.2 we give a rough performance comparison of Pierre, the Pierre protocol modified to use hexagonal tiling (which we call Pierre*), Wilfrid using hexagonal tiling, and $\mathcal{NFP}$-II. We assume that the protocols are implemented in a multiplicative group in the discrete log setting. We only count exponentiation in the underlying group G as that is the most computationally intensive part. We further assume that $|\mathcal{L}| = 2^{40}$ and $|G| = 2^{256}$. The values in Table 6.2 follow from our arguments in Section 6.3.1, 6.3.2, 6.3.3 and 6.3.4.

All the protocols require two communication steps and the messages are elements of G. Note that Pierre and Wilfrid both require that the parties establish a communication channel that is both confidential and authenticated prior to running the protocol and uses homomorphic encryption, while $\mathcal{NFP}$-II only requires an authenticated channel and does not use any other cryptographic primitive.

	$\mathcal{I} \to \mathcal{R}$		$\mathcal{R} \to \mathcal{I}$		decide whether nearby:
	# exps	message size	# exps	message size	# exps by $\mathcal{I}$
Pierre	6	6	8.62	6	≤ 3
Pierre*	6	6	6.62	4	≤ 2
Wilfrid	16	16	4.18	2	1
$\mathcal{NFP}$-II	7	7	8	8	≤ 7

Table 6.2: Performance comparison between Pierre, Wilfrid and $\mathcal{NFP}$-II.

In terms of overall performance, Pierre* is currently the best choice in this case, while $\mathcal{NFP}$-II compares favorably with Wilfrid. In $\mathcal{NFP}$-II we can further reduce the message size from $\mathcal{R}$ to $\mathcal{I}$ by individually hashing the group elements $h_0, \ldots, h_6$ using a standard hash function and sending the hash digests. Such an optimization is not possible for the other protocols as the messages consist of ciphertexts of the underlying homomorphic encryption scheme.

Remark 6.3.1. In Pierre, a malicious responder can always fool the initiator into believing that they are in nearby locations. This may not be a problem as long as $\mathcal{I}$ can physically check that $\mathcal{R}$ is not actually in the location where she is supposed to be. Nor is this an issue for the security of $\mathcal{I}$ as $\mathcal{R}$ cannot gain any information about her location. In fact our security model does not account for this kind of dishonest responder. However, our protocol and Wilfrid appear to resist such behavior.

6.4 Concluding remarks

We formulated a definition and security model of the nearby friend problem in the context of location-based services. We proposed a protocol that efficiently solves the problem in the proposed security model. Our approach is distinct from the previous solutions to this problem as it does not depend on any other cryptographic primitive and requires only an authenticated link. We believe that the basic primitive we proposed will be useful in other contexts.

Chapter 7

Future Work

We finish this thesis by stating some open problems and directions for future research.

In Chapters 2, 3 and 4 we discussed several exponentiation algorithms that work with the compressed representations of elements and compared their performance. In our comparisons, we ignored, for example, the cost of squaring in characteristic-two fields, the cost of cubing in characteristic-three fields and the costs of (torus-based) compression/decompression. It would be desirable to implement the algorithms to verify their relative efficiency.

In Chapter 4 we showed how to compress elements of $G \subset G_{q,4} \subset \mathbb{F}_{q^4}^*$ of orders $q \pm \sqrt{2q} + 1$, where $q = 2^m$, m is odd, by a factor of 4. Our technique allows us to compress elements of subgroups $G_\ell, G_{\bar{\ell}} \subset G_{q,12} \subset \mathbb{F}_{q^{12}}^*$ of orders $\ell = q^2 - t_1 + q - t_2 + 1$ and $\bar{\ell} = q^2 + t_1 + q + t_2 + 1$, where $t_1 = \sqrt{2q^3}$, $t_2 = \sqrt{2q}$ and $q = 2^m$, m is odd, by a factor of 4 because

$$\begin{aligned} q^6 + 1 &\equiv 0 \pmod{\ell}, \\ (q^3 - \sqrt{2q^3} + 1) &\equiv 0 \pmod{\ell}, \end{aligned}$$

and

$$\begin{aligned} q^6 + 1 &\equiv 0 \pmod{\bar{\ell}}, \\ (q^3 + \sqrt{2q^3} + 1) &\equiv 0 \pmod{\bar{\ell}}. \end{aligned}$$

However, the optimal compression factor is 6 since $|\mathbb{F}_{q^{12}}|/|G_\ell| \approx |\mathbb{F}_{q^{12}}|/|G_{\bar{\ell}}| \approx 6$. It would be interesting to achieve the compression factor 6 in G_ℓ and $G_{\bar{\ell}}$ as these groups appear as the images of bilinear pairing functions defined on supersingular hyperelliptic curves with embedding degree 12.

In Chapter 6 we proposed new protocols to solve the nearby friend problem. The nearby friend problem can be viewed as a concrete instantiation of the private matching

problem. The latter problem consists of a large class of problems such as *(multi-party) private set intersection problem, cardinality set intersection problem, private set disjointness test*. Since our protocols do not depend on any other cryptographic primitive it would be interesting to investigate if they can be extended to solve other private matching problems and compare them with the existing protocols. For example, $\mathcal{NFP}$-I can be used to determine whether two singleton sets are equal or not. Similarly, its hierarchical version discussed in Chapter 6.3 can be used to find the intersection of two ordered lists. It would be more challenging to examine whether our approach can be extended to solve the private set intersection problem or whether it is applicable in the multi-party setting. One naive approach to solve the two-party set intersection problem is to modify $\mathcal{NFP}$-I as in Figure 7.1. We call this new protocol the $\mathcal{NFP}$-2PSI protocol.

PUBLIC INFORMATION: A description of the set of elements $\mathcal{L}$, a multiplicative group $G = \langle g \rangle$ of prime order p, and an injective function $f : \mathcal{L} \to G^*$.
INPUT: The initiator $\mathcal{I}$'s input is her secret set $\mathcal{L}_\mathcal{I} = (L_{I_0}, L_{I_1}, \ldots, L_{I_{m-1}})$ and the responder $\mathcal{R}$'s input is his secret set $\mathcal{L}_\mathcal{R} = (L_{R_0}, L_{R_1}, \ldots, L_{R_{n-1}})$.

1. For $0 \leq i \leq m-1$, $\mathcal{I}$ computes $g_{\mathcal{I},i} = f(L_{I_i})$, chooses $\alpha \in_R \mathbb{Z}_p^*$, and computes $a_i = g_{\mathcal{I},i}^{\alpha}$. $\mathcal{I}$ sets $\mathcal{L}_I \cap \mathcal{L}_R \leftarrow \emptyset$ and sends $(a_0, a_1, \ldots, a_{m-1})$ to $\mathcal{R}$.

2. Upon receiving $(a_0, a_1, \ldots, a_{m-1})$ from $\mathcal{I}$, $\mathcal{R}$ first computes $g_{\mathcal{R},j} = f(L_{R_j})$ for $0 \leq j \leq n-1$, chooses $\beta \in_R \mathbb{Z}_p^*$, and then computes $b_j = g_{\mathcal{R},j}^{\beta}$ and $h_i = a_i^{\beta}$ for $0 \leq i \leq m-1$. $\mathcal{R}$ sends to $\mathcal{I}$ the tuple $(b_0, b_1, \ldots, b_{n-1}, h_0, h_1, \ldots, h_{m-1})$.

3. Upon receiving $(b_0, b_1, \ldots, b_{n-1}, h_0, h_1, \ldots, h_{m-1})$ from $\mathcal{R}$, $\mathcal{I}$ compares m and n. If $n \leq m$ then $\mathcal{I}$ computes $h'_j = b_j^{\alpha}$ for $0 \leq j \leq n-1$, then if $h_i = h'_j$ for some $0 \leq i \leq m-1$ and $\mathcal{I}$ sets $\mathcal{L}_I \cap \mathcal{L}_R \leftarrow (\mathcal{L}_I \cap \mathcal{L}_R) \cup L_{\mathcal{I}_i}$. If $m < n$ then $\mathcal{I}$ computes $b'_i = h_i^{\alpha^{-1}}$ for $0 \leq i \leq m-1$, and if $b_j = b'_i$ for some $0 \leq j \leq n-1$ then $\mathcal{I}$ sets $\mathcal{L}_I \cap \mathcal{L}_R \leftarrow (\mathcal{L}_I \cap \mathcal{L}_R) \cup L_{\mathcal{I}_j}$.

Figure 7.1: The $\mathcal{NFP}$-2PSI protocol.

In $\mathcal{NFP}$-2PSI, $\mathcal{I}$ is required to perform m exponentiations in the first step and send m group elements. $\mathcal{R}$ is required to perform $m+n$ exponentiations and send $m+n$ group elements. To determine the intersection $\mathcal{L}_\mathcal{I} \cap \mathcal{L}_\mathcal{R}$, $\mathcal{I}$ has to perform $\min\{m, n\}$ exponentiations in the last step. In Table 7.1, we give a rough comparison of FNP04 (see Chapter 6.3.1) and $\mathcal{NFP}$-2PSI. According to Table 7.1, $\mathcal{NFP}$-2PSI is better than FNP04 in terms of computational complexity and communication bandwidth. However, the security of the $\mathcal{NFP}$-2PSI protocol should be analyzed carefully as the security arguments for the $\mathcal{NFP}$-I protocol do not seem to apply directly in this case.

	$\mathcal{I} \to \mathcal{R}$		$\mathcal{R} \to \mathcal{I}$		compute $\mathcal{L}_\mathcal{I} \cap \mathcal{L}_\mathcal{R}$:
	# exps	message size	# exps	message size	# exps by Alice
FNP04	$2(m+1)$	$2(m+1)$	$2n + 2mn(\log_2 \lvert\mathcal{L}\rvert / \log_2 \lvert G\rvert)$	$2n$	n
$\mathcal{NFP}$-2PSI	m	m	$m+n$	$m+n$	$\min\{m,n\}$

Table 7.1: Performance comparison between FNP04 and $\mathcal{NFP}$-2PSI when implemented in a group G where the discrete log problem is assumed to be hard. Both protocols require two communication steps. The message size is the number of elements of G.

References

[1] M. Abdalla and D. Pointcheval. Interactive Diffie-Hellman assumptions with applications to password-based authentication. *Financial Cryptography and Data Security – FC 2005, Lecture Notes in Computer Science*, 3570:341–356, 2005.

[2] O. Ahmadi, D. Hankerson, and A. Menezes. Software implementation of arithmetic in $\mathbb{F}_{3^m}$. *International Workshop on Arithmetic of Finite Fields – WAIFI 2007, Lecture Notes in Computer Science*, 4547:85–102, 2007.

[3] A. Amir, A. Efrat, J. Myllymaki, L. Palaniappan, and K.Wampler. Buddy tracking - efficient proximity detection among mobile friends. *Pervasive and Mobile Computing*, 3:489–511, 2007.

[4] A. Antipa, D. Brown, A. Menezes, R. Struik, and S. Vanstone. Validation of elliptic curve public keys. *Public Key Cryptography – PKC 2003, Lecture Notes in Computer Science*, 2567:211–223, 2002.

[5] R. Avanzi, H. Cohen, C. Docke, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. *Handbook of elliptic and hyperelliptic curve cryptography*. Chapman & Hall/CRC, FL, USA, 2005.

[6] P. Barreto, H. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing based cryptosystems. *Advances in Cryptology – CRYPTO 2002, Lecture Notes in Computer Science*, 2442:354–369, 2002.

[7] P. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. *Selected Areas in Cryptography – SAC 2005, Lecture Notes in Computer Science*, 3897:319–331, 2006.

[8] D. Bernstein and T. Lange. Faster addition and doubling on elliptic curves. *Advances in Cryptology – ASIACRYPT 2007, Lecture Notes in Computer Science*, 4833:29–50, 2008.

[9] I. Biehl, B. Meyer, and V. Müller. Differential fault analysis on elliptic curve cryptosystems. *Advances in Cryptology – CRYPTO 2000, Lecture Notes in Computer Science*, 1880:131–146, 2000.

[10] D. Bleichenbacher, W. Bosma, and A. Lenstra. Some remarks on Lucas-based cryptosystems. *Advances in Cryptology – CRYPTO'95, Lecture Notes in Computer Science*, 936:306–316, 1995.

[11] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal on Computing*, 32:586–615, 2003. Earlier version appeared in the proceedings of CRYPTO 2001.

[12] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17:297–319, 2004. Earlier version appeared in the proceedings of ASIACRYPT 2001.

[13] A. Brouwer, R. Pellikaan, and E. Verheul. Doing more with fewer bits. *Advances in Cryptology – ASIACRYPT '99, Lecture Notes in Computer Science*, 1716:321–332, 1999.

[14] D. Cantor. Computing the jacobian of a hyperelliptitc curve. *Mathematics of Computation*, 48:95–101, 1987.

[15] G. Cardona, E. Nart, and J. Pujolàs. Curves of genus two over fields of even characteristic. *Mathematische Zeitschrift*, 250:177–201, 2005.

[16] D. Cash, E. Kiltz, and V. Shoup. The twin Diffie-Hellman problem and applications. *Advances in Cryptology – EUROCRYPT 2008, Lecture Notes in Computer Science*, 4965:127–145, 2008. Full version available at `http://eprint.iacr.org/2008/067`.

[17] M. Cenk and F. Özbudak. Isomorphism classes of ordinary elliptic curves over fields of characteristic 3. *Mathematical Methods in Engineering*, Part 3:151–158, 2007.

[18] S. Chatterjee, K. Karabina, and A. Menezes. A new protocol for the nearby friend problem. *Twelfth IMA International Conference on Cryptography and Coding, Lecture Notes in Computer Science*, 5921:236–251, 2009.

[19] L. Chen, Z. Cheng, and N. Smart. Identity-based key agreement protocols from pairings. *International Journal of Information Security*, 6:213–241, 2007.

[20] D. Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE Transactions on Information Theory*, 30:587–594, 1984.

[21] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election. *Advances in Cryptology – EUROCRYPT '97, Lecture Notes in Computer Science*, 1233:103–118, 1997.

[22] Y. Deng and M. Liu. Counting isomorphism classes of pointed hyperelliptic curves of genus 4 over finite fields with odd characteristic. *European Journal of Combinatorics*, 29:1436–1448, 2008.

[23] T. Dierks and E. Rescorla. The transport layer security (TLS)protocol. Technical report, Version 1.1 1483 RFC 4346, Internet Engineering Task Force, 2006. Available at `http://www.ietf.org/rfc/rfc4346.txt`.

[24] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–65, 1976.

[25] W. Du and M. Atallah. Secure multi-party computation problems and their applications: a review and open problems. *Proceedings of the 2001 Workshop on New Security Paradigms – NSPW 2001*, 11–20, 2001.

[26] H. Edwards. A normal form for elliptic curves. *Bulletin of the American Mathematical Society*, 44:393–422, 2007.

[27] L. Encinas and J. Masqué. Isomorphism classes of hyperelliptic curves of genus 2 in characteristic 5. Technical Report CORR 2002-07, University of Waterloo, 2002. Available at `http://www.cacr.math.uwaterloo.ca/techreports/2002/corr2002-07.ps`.

[28] L. Encinas, A. Menezes, and J. Masqué. Isomorphism classes of genus-2 hyperelliptic curves over finite fields. *Applicable Algebra in Engineering, Communication and Computing*, 13:57–65, 2002.

[29] R. Deng, F. Bao and H. Zhu. Variations of Diffie-Hellman problem. *Information and Communications Security – ICICS 2003, Lecture Notes in Computer Science*, 2836:301–312, 2003.

[30] FIPS 186-2. *Digital Signature Standard (DSS).* Federal Information Processing Standards Publication 186-2, National Institute of Standards and Technology, 2000.

[31] FIPS 186-3. *Digital Signature Standard (DSS).* Federal Information Processing Standards Publication 186-3, National Institute of Standards and Technology, 2009.

[32] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. *Advances in Cryptology – EUROCRYPT 2004, Lecture Notes in Computer Science*, 3027:1–19, 2004.

[33] P. Gaudry and É. Schost. Construction of secure random curves of genus 2 over prime fields. *Advances in Cryptology – EUROCRYPT 2004, Lecture Notes in Computer Science*, 3027:239–256, 2004.

[34] K. Giuliani and G. Gong. Analogues to the Gong-Harn and XTR cryptosystems. Technical Report CORR 2003-34, University of Waterloo, 2003. Available at `http://www.cacr.math.uwaterloo.ca/techreports/2003/corr2003-34.ps`.

[35] K. Giuliani and G. Gong. New LFSR-based cryptosystems and the trace discrete log problem (Trace-DLP). *Sequences and Their Applications – SETA 2004, Lecture Notes In Computer Science*, 3486:298–312, 2004.

[36] S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. *ACM Symposium on Theory of Computing – STOC 1982, Association for Computing Machinery*, 365–377, 1982.

[37] G. Gong and L. Harn. Public-key cryptosystems based on cubic finite field extensions. *IEEE Transactions on Information Theory*, 45:2601–2605, 1999.

[38] D. Gordon. Discrete logarithms in $GF(p)$ using the number field sieve. *SIAM Journal on Discrete Mathematics*, 6:124–138, 1993.

[39] R. Granger, D. Page, and M. Stam. On small characteristic algebraic tori in pairing-based cryptography. *LMS Journal of Computation and Mathematics*, 9:64–85, 2004.

[40] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to elliptic curve cryptography*. Springer-Verlag, NY, USA, 2004.

[41] R. Hartshorne. *Algebraic Geometry*. Springer-Verlag, NY, USA, 1997.

[42] D. Husemöller. *Elliptic Curves*. Springer-Verlag, NY, USA, 1987.

[43] T. Icart. How to hash on elliptic curves. *Advances in Cryptology – CRYPTO 2009, Lecture Notes in Computer Science*, 5677:303–316, 2009.

[44] D. Jablon. Strong password-only authenticated key exchange. *ACM SIGCOMM Computer Communication Review*, 26:5–26, 1996.

[45] M. Jakobsson and M. Yung. Proving without knowing: On oblivious, agnostic and blindfolded provers. *Advances in Cryptology – CRYPTO '96, Lecture Notes in Computer Science*, 1109:186–200, 1996.

[46] E. Jeong. Isomorphism classes of hyperelliptic curves of genus 3 over finite fields. Cryptology ePrint Archive, Report 2003/251, 2003. Available at `http://eprint.iacr.org/2003/251`.

[47] A. Joux. A one round protocol for tripartite Diffie-Hellman. *Algorithmic Number Theory 4th International Symposium – ANTS IV, Lecture Notes in Computer Science*, 1838:385–393, 2000.

[48] A. Joux and R. Lercier. Improvements to the general number field sieve for discrete logarithms in prime fields – A comparison with the Gaussian integer method. *Mathematics of Computation*, 72:953–967, 2003.

[49] K. Karabina. Double-exponentiation in factor-4 groups and its applications. *Twelfth IMA International Conference on Cryptography and Coding, Lecture Notes in Computer Science*, 5921:336–350, 2009.

[50] K. Karabina. Factor-4 and 6 compression of cyclotomic subgroups of $\mathbb{F}^*_{2^{4m}}$ and $\mathbb{F}^*_{3^{6m}}$. *Journal of Mathematical Cryptology*, to appear.

[51] K. Karabina and B. Ustaoglu. Invalid-curve attacks on (hyper)elliptic curve cryptosystems. *Advances in Mathematics of Communications*, to appear.

[52] L. Kissner and D. Song. Privacy-preserving set operations. *Advances in Cryptolgy – CRYPTO 2005, Lecture Notes in Computer Science*, 3621:241–257, 2005.

[53] N. Koblitz. An elliptic curve implementation of the finite field digital signature algorithm. *Advances in Cryptology – CRYPTO '98, Lecture Notes in Computer Science*, 1462:327–337, 1998.

[54] G. Køien and V. Oleshchuk. Location privacy for cellular systems; analysis and solution. *Privacy Enhancing Technology Workshop – PET 2005, Lecture Notes in Computer Science*, 3856:40–58, 2005.

[55] H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. *Advances in Cryptology – CRYPTO 2005, Lecture Notes in Computer Science*, 3621:546–566, 2005. Full version available at `http://eprint.iacr.org/2005/176`.

[56] S. Kunz-Jacques, G. Martinet, G. Poupard, and J. Stern. Cryptanalysis of an efficient proof of knowledge of discrete logarithm. *Public Key Cryptography – PKC 2006, Lecture Notes in Computer Science*, 3958:27–43, 2006.

[57] B. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. *Provable Security: First International Conference, ProvSec 2007, Lecture Notes in Computer Science*, 4784:1–16, 2007.

[58] A. Lenstra. Unbelievable security. Matching AES security using public key systems. *Advances in Cryptology – ASIACRYPT 2001, Lecture Notes in Computer Science*, 2248:67–86, 2001.

[59] A. Lenstra and E. Verheul. The XTR public key system. *Advances in Cryptology – CRYPTO 2000, Lecture Notes in Computer Science*, 1880:1–19, 2000.

[60] H. Lenstra. Factoring integers with elliptic curves. *Annals of Mathematics*, 126:649–673, 1987.

[61] H. Lenstra, J. Pila, and C. Pomerance. A hyperelliptic smoothness test, II. *Proceedings of the London Mathematical Society*, 84:105–146, 2002.

[62] C. Lim and P. Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. *Advances in Cryptology – CRYPTO'97, Lecture Notes in Computer Science*, 1294:249–263, 1997.

[63] P. Lockhart. On the discriminant of a hyperelliptic curve. *Transactions of the American Mathematical Society*, 342:729–752, 1994.

[64] A. Menezes. Another look at HMQV. *Journal of Mathematical Cryptology*, 1:47–64, 2007.

[65] A. Menezes and B. Ustaoglu. On the importance of public-key validation in the MQV and HMQV key agreement protocols. *Progress in Cryptology – INDOCRYPT 2006, Lecture Notes in Computer Science*, 4329:133–147, 2006.

[66] A. Menezes and B. Ustaoglu. On reusing ephemeral keys in Diffie-Hellman key agreement protocols. *International Journal of Applied Cryptography*, to appear.

[67] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of applied cryptography*. CRC Press, NY, USA, 1997.

[68] P. Montgomery. Evaluating recurrences of form $X_{m+n} = f(X_m, X_n, X_{m-n})$ via Lucas chains, December 13, 1983; Revised March, 1991 and January, 1992. Available at `www.cwi.nl/ftp/pmontgom/Lucas.ps.gz`.

[69] J. Muir and D. Stinson. Minimality and other properties of the width-w nonadjacent form. *Mathematics of Computation*, 75:369–384, 2005.

[70] W. Müller and W. Nöbauer. Some remarks on public-key cryptosystems. *Studia Scientiarum Mathematicarum Hungarica*, 16:71–76, 1981.

[71] E. Nart. Counting hyperelliptic curves. *Advances in Mathematics*, 221:774–787, 2009.

[72] K. Nyberg and A. Rueppel. Message recovery for signature schemes based on the discrete logarithm problem. *Designs, Codes and Cryptography*, 7:61–81, 1996.

[73] J. Pollard. Monte Carlo methods for index computation mod p. *Mathematics of Computation*, 32:918–924, 1978.

[74] K. Rubin and A. Silverberg. Torus-based cryptography. *Advances in Cryptology – CRYPTO 2003, Lecture Notes in Computer Science*, 2729:349–465, 2003.

[75] R. Sakai, T. Ohgishi, and M. Kasahara. Cryptosystems based on pairings. *Symposium on Cryptography and Information Security – SCIS 2000, Okinawa, Japan*, 26–28, 2000.

[76] M. Scott. Authenticated ID-based key exchange and remote log-in with simple token and PIN number, 2002. Available at `http://eprint.iacr.org/2002/164`.

[77] M. Scott and P. Barreto. Compressed pairings. *Advances in Cryptology – CRYPTO 2004, Lecture Notes In Computer Science*, 3152:140–156, 2004.

[78] M. Shirase, D. Han, Y. Hibin, H. Kim, and T. Takagi. A more compact representation of XTR cryptosystem. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E91-A:2843–2850, 2008.

[79] V. Shoup. A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic. *International Symposium on Symbolic and Algebraic Computation – ISSAC '91, Association for Computing Machinery*, 14–21, 1991.

[80] V. Shoup. Lower bounds for discrete logarithms and related problems. *Advances in Cryptology – EUROCRYPT '97, Lecture Notes in Computer Science*, 1233:256–266, 1997.

[81] J. Silverman. *The Arithmetic of Elliptic Curves*. Springer-Verlag, NY, USA, 1986.

[82] P. Smith and M. Lennon. LUC: A new public key system. *Proceedings of the Ninth IFIP International Symposium on Computer Security*, 103–117, 1993.

[83] P. Smith and C. Skinner. A public-key cryptosystem and a digital signature system based on the Lucas function analogue to discrete logarithms. *Advances in Cryptology – ASIACRYPT '94, Lecture Notes In Computer Science*, 917:357–364, 1994.

[84] M. Stam and A. Lenstra. Speeding up XTR. *Advances in Cryptology – ASIACRYPT 2001, Lecture Notes in Computer Science*, 2248:125–143, 2001.

[85] M. Szydlo. A note on chosen-basis decisional Diffie-Hellman assumptions. *Financial Cryptography and Data Security – FC 2005, Lecture Notes in Computer Science*, 4107:166–170, 2006.

[86] T. Takagi, S. Yen, and B. Wu. Radix-r non-adjacent form. *Information Security – ISC 2004, Lecture Notes In Computer Science*, 3225:99–110, 2004.

[87] B. Ustaoglu. Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. *Designs, Codes and Cryptography*, 46:329–342, 2008.

[88] M. van Dijk, R. Granger, D. Page, K. Rubin, A. Silverberg, M. Stam, and D. Woodruff. Practical cryptography in high dimensional tori. *Advances in Cryptology – EUROCRYPT 2005, Lecture Notes in Computer Science*, 3494:234–250, 2005.

[89] M. van Dijk and D. Woodruff. Asymptotically optimal communication for torus-based cryptography. *Advances in Cryptology – CRYPTO'2004, Lecture Notes in Computer Science*, 3152:151–178, 2004.

[90] S. Wang, J. Min, and B. Yi. Location based services for mobiles: Technologies and standards. *IEEE International Conference on Communication – ICC*, 2008.

[91] B. Waters. Efficient identity-based encryption without random oracles. *Advances in Cryptology – EUROCRYPT 2005, Lecture Notes in Computer Science*, 3494:114–127, 2005.

[92] A. Weil. L'arithmétique sur les courbes algébriques. *Acta Mathematica*, 52:281–315, 1928.

[93] G. Zhong. Distributed approaches for location privacy. Master's thesis, University of Waterloo, 2008.

[94] G. Zhong, I. Goldberg, and U. Hengartner. Louis, Lester and Pierre: Three protocols for location privacy. *Privacy Enhancing Technologies – PET 2007, Lecture Notes in Computer Science*, 4776:62–76, 2007.