Towards practical non-interactive public-key cryptosystems using non-maximal imaginary quadratic orders

Detlef Hühnlein (huehnlein@secunet.de)
secunet Security Networks AG
Mergenthalerallee 77-81, D-65760 Eschborn, Germany

Michael J. Jacobson, Jr. (jacobs@cs.umanitoba.ca)
Department of Computer Science
University of Manitoba
Winnipeg, Manitoba, Canada, R3T 2N2

Damian Weber (weber@informatik.fh-trier.de)
Fachhochschule Trier
Schneidershof, D-54293 Trier, Germany

Abstract. We present a new non-interactive public-key distribution system based on the class group of a non-maximal imaginary quadratic order $Cl(\Delta_p)$. The main advantage of our system over earlier proposals based on $(\mathbb{Z}/n\mathbb{Z})^*$ [26, 28] is that embedding id information into group elements in a cyclic subgroup of the class group is easy (straight-forward embedding into prime ideals suffices) and secure, since the entire class group is cyclic with very high probability. Computational results demonstrate that a key generation center (KGC) with modest computational resources can set up a key distribution system using reasonably secure public system parameters.

In order to compute discrete logarithms in the class group, the KGC needs to know the prime factorization of $\Delta_p = \Delta_1 p^2$. We present an algorithm for computing discrete logarithms in $Cl(\Delta_p)$ by reducing the problem to computing discrete logarithms in $Cl(\Delta_1)$ and either $\mathbb{F}_p^*$ or $\mathbb{F}_{p^2}^*$. Our algorithm is a special case of that in the more general setting of ray class groups [5], but we present it in terms of ideals of quadratic orders without using class field theoretic language, and we prove – for arbitrary non-maximal orders – that the reduction to discrete logarithms in the maximal order and a small number of finite fields has polynomial complexity if the factorization of the conductor is known.

Keywords: discrete logarithm, non-maximal imaginary quadratic order, identity based cryptography, non-interactive cryptography

1. Introduction

Public-key cryptography is undoubtedly one of the core techniques used to enable authentic, non-repudiable and confidential communication. However, a general problem inherent in public-key systems is that one needs to ensure the authenticity of a given public key. The most common way to solve this problem is to introduce a trusted third party, called a *Certification Authority* (CA), which issues certificates for public keys[1]. While this approach is widely used in practice, it would be desirable to have an immediate binding between an identity ID_B and its corresponding public key $\mathfrak{b}$, which allows one to avoid the tedious verification of certificates. This leads to the notion of *identity based cryptosystems*, as proposed by Shamir [36]. For signature schemes, the public key $\mathfrak{b}$ is only needed when a user receives a signed message, and thus it is tolerable that the public key $\mathfrak{b}$ is derived from ID_B *and* some identity-specific system parameter SP_B, which can easily be appended in this case. However, in order to achieve *non-interactive* public-key encryption and key distribution schemes, it is necessary that the knowledge of ID_B *alone* is sufficient to derive the public key $\mathfrak{b}$. This type of scheme was first proposed by Maurer and Yacobi [26]. They proposed setting up a discrete logarithm based system in $G = (\mathbb{Z}/n\mathbb{Z})^*$, where $n = p_1 \cdots p_r$, p_i prime, such that only a key generation center (KGC) which knows the factorization of n is able to compute discrete logarithms in G. However, as we will see in Section 2, this approach has a number of drawbacks which render such a scheme impractical.

In this paper we show that using the class group $Cl(\Delta_p)$ of a non-maximal imaginary quadratic order is much better suited for this purpose. As in the original scheme, the KGC knows some trapdoor information which enables it to compute discrete logarithms, while for anybody else the discrete logarithm problem is (assumed to be) intractable. We begin by generalizing the recent result from [15], valid for the very special case of totally non-maximal orders with prime discriminant, to arbitrary non-maximal imaginary quadratic orders. The resulting algorithm reduces the problem of discrete logarithm computation in the class group of a non-maximal order to computing discrete logarithms in the much smaller class group of the corresponding maximal order and a small number of finite fields. This algorithm is a special case of that for computing discrete logarithms in ray class groups [5], but we present it in the framework of maximal and non-maximal orders rather than ray class groups. In addition, we prove that the reduction to discrete logarithms in the corresponding maximal order and finite fields is of polynomial complexity. These results are then applied to set up a more practical non-interactive scheme using $Cl(\Delta_p)$.

As noted above there are a few advantages to our approach. Unlike the case of $(\mathbb{Z}/n\mathbb{Z})^*$, it is heuristically easy to find class groups $Cl(\Delta_p)$ which are *cyclic*, and hence the embedding of an identity ID_B into a group element $\mathfrak{b}$, for which the discrete logarithm exists, is straightforward. As the results from [27, 25] demonstrate it seems to be no trivial

[1] We assume throughout this work that Alice (A) wants to encrypt a message $m \in \mathbb{Z}_{>0}$ intended for Bob (B). We denote Bob's unique identity, for example his email-address, by ID_B and his public key by $\mathfrak{b}$.

task to find an embedding into a subgroup of $(\mathbb{Z}/n\mathbb{Z})^*$ which does not facilitate factoring n. In fact, the only secure embedding method for $(\mathbb{Z}/n\mathbb{Z})^*$ seems to restrict n to having *only two* large prime factors p_1 and p_2, and the workload for the KGC is consequently very high. Furthermore, since one chooses $p_i - 1$ smooth and uses Pohlig-Hellman's simplification together with Shank's Baby-Step Giant-Step algorithm, the time needed for generating k user keys is proportional to k.

In contrast, we use two different subexponential algorithms for the key generation. After the initial computation of relations over the factor bases, the workload for each individual key generation is very modest. For the computation of discrete logarithms in $Cl(\Delta_1)$ we use an analogue of the Self-Initializing Quadratic Sieve (SIQS) factoring algorithm [16, 17] and for the computation of discrete logarithms in $\mathbb{F}_p^*$ we use the Special Number Field Sieve, which recently was used for the solution of McCurley's challenge [38].

This paper is organized as follows: In Section 2 we briefly recall previous proposals for non-interactive public-key cryptosystems. In Section 3 we provide the necessary background and notation for non-maximal imaginary quadratic orders. The next section briefly summarizes the current state-of-the art algorithms for computing discrete logarithms in finite fields and maximal imaginary quadratic orders, the two necessary ingredients for our reduction of discrete logarithm computation in non-maximal orders. Section 5 contains the discrete logarithm algorithm for arbitrary non-maximal imaginary quadratic orders. In Section 6 we present our new non-interactive public-key cryptosystem, followed by computational examples in Section 7.

2. Previous proposals of non-interactive cryptosystems

Although the paradigm of identity based cryptography was already introduced by Shamir in 1984 [36], it seems that Maurer and Yacobi [26] were the first to propose a *non-interactive* identity based public-key cryptosystem in which Bob's public key $\mathfrak{b}$ can be derived efficiently, solely from his public identity information ID_B, by computing a publicly-known embedding function $\mathfrak{b} = f(ID_B)$. The main idea is to use an (ideally cyclic) group G (generated by $\mathfrak{g}$) in which exponentiation is not only a one-way function but a *trapdoor one-way function.* The KGC knows the trapdoor information and hence is able to compute discrete logarithms in G. Thus, the KGC computes Bob's private key b such that $\mathfrak{g}^b = \mathfrak{b} = f(ID_B)$. The KGC hands over the secret key b to Bob, who can use this key in a conventional ElGamal or Diffie-Hellman setup. As soon as all users are equipped with their corresponding secret key, the KGC can destroy the trapdoor information and may cease to exist.

One approach to set up such a non-interactive cryptosystem would be to use the group $G = (\mathbb{Z}/n\mathbb{Z})^*$, where $n = p_1 \cdots p_r$ is the product of r different primes. The KGC generates n such that factoring it is hard and publishes n, while it keeps the prime factors secret. However, it is well-known that $(\mathbb{Z}/n\mathbb{Z})^*$ is cyclic if and only if $n \in \{2, 4, 2p^k, p^k\}$ for an odd prime p and $k \in \mathbb{Z}_{>0}$. Since we require that factoring n is hard we obviously cannot use such a modulus n, and consequently, we cannot guarantee that the discrete logarithm for some $\mathfrak{b} = ID_B$ to a universal base element $\mathfrak{g}$ exists. Therefore one needs to apply a more sophisticated embedding function which maps an identity ID_B into a cyclic subgroup of $(\mathbb{Z}/n\mathbb{Z})^*$.

Maurer and Yacobi proposed choosing $\mathfrak{g}$ to be a simultaneous primitive root of all finite fields $\mathbb{F}_{p_1}^*, \cdots, \mathbb{F}_{p_r}^*$ and the following embeddings to guarantee the existence of discrete logarithms to the base $\mathfrak{g}$:

1. *Squaring method*
   In [26] they proposed using $f(ID_B) := ID_B^2 \bmod n$. However, later on [27] they recognized that this method is extremely vulnerable, because *a single user* can find a non-trivial factor of n with probability $1 - 2^{-r+1} \geq 1/2$. To avoid this weakness, they proposed that the KGC masks all secret keys with a fixed $t \in (\mathbb{Z}/\phi(n)\mathbb{Z})^*$ and perform all other computations as usual. However, in [25] it was shown that with high probability this proposed masking does not prevent factoring n when two users mutually disclose their secret key. Thus, the squaring method should not be used, due to security reasons.

2. *Legendre Symbol method*
   In [26] they also propose using the special case of $n = p_1 p_2$ as "an alternative though less practical approach ... for the sake of completeness." They initially proposed taking the smallest number $\mathfrak{b} \geq ID_B$ with $(\mathfrak{b}/n) = 1$ as the public key. In [27] they refined this embedding-function, and proposed using $p_1 \equiv 3 \pmod 8$ and $p_2 \equiv 7 \pmod 8$ such that $(2/n) = -1$ and

$$\mathfrak{b} = f(ID_B) = \begin{cases} ID_B & \text{if } \left(\frac{ID_B}{n}\right) = 1 \\ 2ID_B & \text{if } \left(\frac{ID_B}{n}\right) = -1 \end{cases}$$

to guarantee the existence of a discrete logarithm. In both cases the method is only feasible[2] for the case $n = p_1 p_2$. They propose choosing $p_1 - 1$ and $p_2 - 1$ fairly[3] smooth and use the algorithm of Pohlig-Hellman to compute the discrete logarithms. But, unlike using subexponential time algorithms for discrete logarithm computation, the computation of every individual logarithm is very expensive. Therefore this approach is too inefficient to be used in practice. Lim and Lee [23] came to a similar conclusion.

It should be noted that Okamoto and Uchiyama [31] proposed an analogous system using the group of points on an anomalous elliptic curve over the ring $(\mathbb{Z}/n\mathbb{Z})$, $n = p_1 p_2$, which seemed to be very practical because the discrete logarithm in this group can be computed in polynomial time if one knows the factorization of n. Unfortunately, they found (prior to publication) that this scheme can be completely broken because one easily finds the factorization of n in this setup.

In [19] Kügler studied the application of a public factor base to obtain practical non-interactive schemes. While the key generation for the KGC can be performed in polynomial time this approach has the severe drawback that every user needs to store a public factor base, which may need more than 1 MByte in a practical setup. Furthermore, the size of the factor base needs to be at least as large as the number of users to prevent an attack by solving a system of linear equations.

3. Background and notation for non-maximal imaginary quadratic orders

The basic notions of imaginary quadratic number fields can be found in [3, 4]. For a more comprehensive treatment of the relationship between maximal and non-maximal orders we refer to [7, 13, 15, 30].

3.1. Maximal imaginary quadratic orders

Let $\Delta \equiv 0, 1 \pmod 4$ be a negative integer whose absolute value is not a square. The quadratic order of discriminant Δ is defined to be

$$\mathcal{O}_\Delta = \mathbb{Z} + \omega\mathbb{Z} \ ,$$

[2] In [27] they also propose a generalization to the case $n = p_1 \cdots p_r$, but this approach is completely impractical since both Alice and Bob need to perform $r - 2$ additional exponentiations and the transmitted key is $r - 2$ times as long.

[3] Since Pollard's $p - 1$ factoring algorithm [32] factors n efficiently if all prime factors of $p_i - 1$ are smooth one should choose the primes q dividing $p_i - 1$ such that $q \geq 2^{40}$.

where

$$\omega = \begin{cases} \sqrt{\frac{\Delta}{4}}, & \text{if } \Delta \equiv 0 \pmod 4 \ , \\ \frac{1+\sqrt{\Delta}}{2}, & \text{if } \Delta \equiv 1 \pmod 4 \ . \end{cases} \tag{1}$$

The standard representation of $\alpha \in \mathcal{O}_\Delta$ is $\alpha = x + y\omega$, where $x, y \in \mathbb{Z}$.

If Δ_1 (or $\Delta_1/4$ if $\Delta \equiv 0 \pmod 4$) is square-free, then $\mathcal{O}_{\Delta_1}$ is the *maximal order* of the quadratic number field $\mathbb{Q}(\sqrt{\Delta_1})$ and Δ_1 is called a fundamental discriminant. The *non-maximal order* of conductor $f > 1$ with non-fundamental discriminant $\Delta_f = \Delta_1 f^2$ is denoted by $\mathcal{O}_{\Delta_f}$. We omit the subscripts to reference arbitrary (fundamental or non-fundamental) discriminants. Because $\mathbb{Q}(\sqrt{\Delta_1}) = \mathbb{Q}(\sqrt{\Delta_f})$ we also omit the subscripts to reference the number field $\mathbb{Q}(\sqrt{\Delta})$. The *standard representation* of an $\mathcal{O}_\Delta$-ideal is

$$\mathfrak{a} = q\left(\mathbb{Z} + \frac{b + \sqrt{\Delta}}{2a}\mathbb{Z}\right) = q(a, b) \ ,$$

where $q \in \mathbb{Q}_{>0}, a \in \mathbb{Z}_{>0}$, $c = (b^2 - \Delta)/(4a) \in \mathbb{Z}$, $\gcd(a, b, c) = 1$ and $-a < b \leq a$. The norm of this ideal is $\mathcal{N}(\mathfrak{a}) = aq^2$. An ideal is called primitive if $q = 1$. The standard representation of a primitive ideal boils down to (a, b). A primitive ideal is called *reduced* if $|b| \leq a \leq c$ and $b \geq 0$ if $a = c$. It can be shown that the norm of a reduced ideal $\mathfrak{a}$ satisfies $\mathcal{N}(\mathfrak{a}) \leq \sqrt{|\Delta|/3}$ and conversely that if $\mathcal{N}(\mathfrak{a}) \leq \sqrt{|\Delta|/4}$ then the ideal $\mathfrak{a}$ is reduced.

The group of invertible $\mathcal{O}_\Delta$-ideals is denoted by $\mathcal{I}_\Delta$. Two ideals $\mathfrak{a}, \mathfrak{b}$ are said to be equivalent if there is a $\gamma \in \mathbb{Q}(\sqrt{\Delta})$, such that $\mathfrak{a} = \gamma\mathfrak{b}$. This equivalence relation is denoted by $\mathfrak{a} \sim \mathfrak{b}$. The set of principal $\mathcal{O}_\Delta$-ideals, i.e., those ideals which are equivalent to $\mathcal{O}_\Delta$, is denoted by $\mathcal{P}_\Delta$. The factor group $\mathcal{I}_\Delta/\mathcal{P}_\Delta$ is called the *class group* of $\mathcal{O}_\Delta$, denoted by $Cl(\Delta)$. The group elements are equivalence classes (denoted by $[\mathfrak{a}]$), and the neutral element is the class of ideals equivalent to $\mathcal{O}_\Delta$. Each equivalence class can be represented uniquely by a reduced ideal. Algorithms for the group operation (multiplication and reduction of ideals) can be found in [4]. $Cl(\Delta)$ is a finite abelian group, and its order is called the *class number* of $\mathcal{O}_\Delta$, denoted by $h(\Delta)$.

3.2. Non-maximal imaginary quadratic orders

Our cryptosystem makes use of the relationship between a non-maximal order of conductor f and its corresponding maximal order. Any non-maximal order can be represented as $\mathcal{O}_{\Delta_f} = \mathbb{Z} + f\mathcal{O}_{\Delta_1}$. If $h(\Delta_1) = 1$, then $\mathcal{O}_{\Delta_f}$ is called a totally non-maximal order. An $\mathcal{O}_\Delta$-ideal $\mathfrak{a}$ is called prime to f if $\gcd(\mathcal{N}(\mathfrak{a}), f) = 1$. It is well-known that all $\mathcal{O}_{\Delta_f}$-ideals prime to the conductor are invertible, and in every ideal equivalence class there is an ideal which is prime to any given integer. We denote the principal $\mathcal{O}_{\Delta_f}$-ideals, which are prime to f by $\mathcal{P}_{\Delta_f}(f)$ and all $\mathcal{O}_{\Delta_f}$-ideals which are prime to f by $\mathcal{I}_{\Delta_f}(f)$. Then there is an isomorphism

$$\mathcal{I}_{\Delta_f}(f)/\mathcal{P}_{\Delta_f}(f) \simeq \mathcal{I}_{\Delta_f}/\mathcal{P}_{\Delta_f} = Cl(\Delta_f) \ , \tag{2}$$

so we can "ignore" the ideals which are not prime to the conductor if we are only interested in the class group $Cl(\Delta_f)$.

There is an isomorphism between the group of $\mathcal{O}_{\Delta_f}$-ideals which are prime to f and the group of $\mathcal{O}_{\Delta_1}$-ideals which are prime to f, denoted by $\mathcal{I}_{\Delta_f}(f)$, and $\mathcal{I}_{\Delta_1}(f)$, respectively.

PROPOSITION 1. *Let $\mathcal{O}_{\Delta_f}$ be an order of conductor f in an imaginary quadratic field $\mathbb{Q}(\sqrt{\Delta})$ with maximal order $\mathcal{O}_{\Delta_1}$.*

(i.) *If $\mathfrak{A} \in \mathcal{I}_{\Delta_1}(f)$, then $\mathfrak{a} = \mathfrak{A} \cap \mathcal{O}_{\Delta_f} \in \mathcal{I}_{\Delta_f}(f)$ and $\mathcal{N}(\mathfrak{A}) = \mathcal{N}(\mathfrak{a})$.*
(ii.) *If $\mathfrak{a} \in \mathcal{I}_{\Delta_f}(f)$, then $\mathfrak{A} = \mathfrak{a}\mathcal{O}_{\Delta_1} \in \mathcal{I}_{\Delta_1}(f)$ and $\mathcal{N}(\mathfrak{a}) = \mathcal{N}(\mathfrak{A})$.*
(iii.) *The map $\varphi : \mathfrak{A} \mapsto \mathfrak{A} \cap \mathcal{O}_{\Delta_f}$ induces an isomorphism $\mathcal{I}_{\Delta_1}(f) \xrightarrow{\sim} \mathcal{I}_{\Delta_f}(f)$. The inverse of this map is $\varphi^{-1} : \mathfrak{a} \mapsto \mathfrak{a}\mathcal{O}_{\Delta_1}$.*

Proof. See [7, Proposition 7.20, p.144]. □

Thus we are able to switch to and from ideals in the maximal and non-maximal orders via the map φ. The algorithms $\mathsf{GoToMaxOrder}(\mathfrak{a}, f)$ to compute φ^{-1} and $\mathsf{GoToNonMaxOrder}(\mathfrak{A}, f)$ to compute φ respectively can be found in [13]. If $\mathfrak{a} = a\mathbb{Z} + (b + \sqrt{\Delta_f})/2\mathbb{Z} = (a, b)$ and $\mathfrak{A} = A\mathbb{Z} + (B + \sqrt{\Delta_1})/2\mathbb{Z} = (A, B)$ are reduced ideals, then these algorithms need $O(\log(|\Delta_1|)^2)$ and $O(\log(|\Delta_f|)^2)$ bit-operations respectively.

It is important to note that the isomorphism φ is between the *ideal groups* $\mathcal{I}_{\Delta_1}(f)$ and $\mathcal{I}_{\Delta_f}(f)$ and *not the class groups*. If, for $\mathfrak{A}, \mathfrak{B} \in \mathcal{I}_{\Delta_1}(f)$ we have $\mathfrak{A} \sim \mathfrak{B}$, it is not necessarily true that $\varphi(\mathfrak{A}) \sim \varphi(\mathfrak{B})$. On the other hand, equivalence *does* hold under φ^{-1}. More precisely we have the following:

PROPOSITION 2. *The isomorphism φ^{-1} induces a* surjective homomorphism
$\phi_{Cl}^{-1} : Cl(\Delta_f) \to Cl(\Delta_1)$, *where* $[\mathfrak{a}] \mapsto [\varphi^{-1}(\mathfrak{a})]$.

Proof. This immediately follows from the short exact sequence:

$$Cl(\Delta_f) \longrightarrow Cl(\Delta_1) \longrightarrow 1$$

(see [30, Theorem 12.9, p. 82]). □

We now focus on the kernel $\mathrm{Ker}(\phi_{Cl}^{-1})$ of this map, which will turn out to be of central importance for the computation of discrete logarithms in $Cl(\Delta_f)$. In particular, we will need to compute discrete logarithms of elements in $\mathrm{Ker}(\phi_{Cl}^{-1})$. Representing elements of $\mathrm{Ker}(\phi_{Cl}^{-1})$ as ideal equivalence classes is completely inadequate for this purpose since we would have to compute discrete logarithms in $Cl(\Delta_f)$. Fortunately, there exists an alternative representation which allows us to reduce the problem of computing discrete logarithms in $\mathrm{Ker}(\phi_{Cl}^{-1})$ to that in a small number of finite fields.

PROPOSITION 3. *The map* $\psi : (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^* \to \mathrm{Ker}(\phi_{Cl}^{-1})$, $[\alpha] \mapsto [\varphi(\alpha\mathcal{O}_{\Delta_1})]$, *is a surjective homomorphism.*

Proof. This is shown in the more comprehensive proof of Theorem 7.24 in [7, p.147]. □

This homomorphism suggests the following representation for ideal classes in the kernel:

DEFINITION 1. *Let* $[\alpha] = [x + y\omega] \in (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*$ *and let* $\mathfrak{a} \sim \varphi(\alpha\mathcal{O}_{\Delta_1})$ *be a reduced* $\mathcal{O}_{\Delta_f}$*-ideal whose equivalence class is in* $\mathrm{Ker}(\phi_{Cl}^{-1})$. *Then the pair* (x, y) *is called a* generator representation *for the equivalence class* $[\mathfrak{a}]$.

DEFINITION 2. *Let* $Im((\mathbb{Z}/f\mathbb{Z})^*)$ *denote the natural embedding of* $(\mathbb{Z}/f\mathbb{Z})^*$ *into* $(\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*$, *i.e.,*

$$\begin{aligned} Im : (\mathbb{Z}/f\mathbb{Z})^* &\mapsto (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^* \\ [x] &\mapsto [(x,x)] \ . \end{aligned}$$

Remark 1. Note that this generator representation (x, y) for the class of $\mathfrak{a}$ is *not unique*. It is easy to see that (kx, ky), $k \in (\mathbb{Z}/f\mathbb{Z})^*$, is also a generator representation for the class of $\mathfrak{a}$. This means that we have $\mathfrak{a} \sim \varphi((x + y\omega)\mathcal{O}_{\Delta_1}) \sim \varphi((kx + ky\omega)\mathcal{O}_{\Delta_1})$. In other words, $\mathrm{Ker}(\phi_{Cl}^{-1}) \cong (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*/Im((\mathbb{Z}/f\mathbb{Z})^*)$, as illustrated by the exact sequence (7.27) in [7, p.147].

Our reduction of the discrete logarithm problem in $Cl(\Delta_f)$ to $Cl(\Delta_1)$ and finite fields requires computing various preimages of elements in $\mathrm{Ker}(\phi_{Cl}^{-1})$ under the map ψ. Algorithm 1 (Std2Gen) accomplishes this task. The algorithm Reduce reduces an ideal $\mathfrak{A}$ given in standard representation and simultaneously computes a reducing number $\gamma \in \mathcal{O}_{\Delta_1}$ of the form $(x + y\sqrt{\Delta_1})/2$ such that $\mathfrak{A}/\gamma$ is reduced (see, for example, [16, Algorithm 2.6, p.16]).

Proof (correctness of Std2Gen). The first step in the routine GoToMaxOrder [13] is to compute an ideal $\mathfrak{a}' \sim \mathfrak{a}$ with $\gcd(\mathcal{N}(\mathfrak{a}'), f) = 1$.

Algorithm 1 Std2Gen

Input: The standard representation (a, b) of a reduced $\mathcal{O}_{\Delta_f}$-ideal $\mathfrak{a} = a\mathbb{Z} + \frac{b+\sqrt{\Delta_f}}{2}\mathbb{Z}$ representing a class in $\mathrm{Ker}(\phi_{Cl}^{-1})$, and the conductor f.
Output: A generator representation (x, y) of the class $[\mathfrak{a}] \in \mathcal{O}_{\Delta_1}$.

```
(a, b) ← GoToMaxOrder(𝔞, f)
(𝔊, γ) ← Reduce(a, b)
if 𝔊 ≁ O_{Δ_1} then
  return('Error! 𝔞 ∉ Ker(φ_Cl^{-1})!')
end if
if Δ_1 ≡ 0 (mod 4) then
  x̄ ← x/2 (mod f)
  ȳ ← y/2 (mod f)
else
  x̄ ← (x − y)/2 (mod f)
  ȳ ← y (mod f)
end if
return((x̄, ȳ))
```

Thus, we obtain the principal ideal $\mathfrak{A} = \gamma\mathcal{O}_{\Delta_1} = \varphi^{-1}(\mathfrak{a}') = a\mathbb{Z} + (b + \sqrt{\Delta_1})/2\mathbb{Z}$ in standard representation. The algorithm Reduce computes $\mathfrak{G} \sim \mathfrak{A}$ such that $\mathfrak{G}$ is reduced, together with $\gamma = (x + y\sqrt{\Delta_1})/2$ such that $\mathfrak{G} = \mathfrak{A}/\gamma$. If $\mathfrak{G} \neq \mathcal{O}_{\Delta_1}$, then $\mathfrak{a}$ cannot be in the kernel $\mathrm{Ker}(\phi_{Cl}^{-1})$ and an error is returned. Otherwise, since $\mathfrak{G} = \mathfrak{A}/\gamma$ and $\mathfrak{G} = \mathcal{O}_{\Delta_1}$ we have $(\gamma) = \mathfrak{A}$, i.e., γ is a generator of the principal ideal $\mathfrak{A}$. Finally, we simply convert γ to the form $x + y\omega$, and since $\gcd(\mathcal{N}(\mathfrak{a}'), f) = \gcd(\mathcal{N}(\mathfrak{A}), f) = \gcd(N(\gamma), f) = 1$, we may apply [10, Lemma 5] and reduce modulo f without leaving the equivalence class of $\mathfrak{a}$. □

PROPOSITION 4. *Std2Gen needs $O(\log(|\Delta_f|)^2)$ bit-operations.*

Proof. Since $\mathfrak{a}$ is reduced, GoToMaxOrder needs $O(\log(|\Delta_f|)^2)$ bit-operations. Since $a = \mathcal{N}(\mathfrak{a}')$, we know by [2] that the reduction, including the computation of γ, also takes $O((\log|\Delta_f|)^2)$ bit-operations. □

4. DLP in $Cl(\Delta)$ and finite fields — state of the art

Let G be a finite abelian (multiplicatively written) group and $\mathfrak{g} \in G$ be a fixed element. Then the *discrete logarithm problem* (DLP) in G for a given $\mathfrak{a}$ is to determine an $a \in \mathbb{Z}$ such that $\mathfrak{g}^a = \mathfrak{a}$, or show that no such a exists.

The best available algorithm for computing discrete logarithms in finite fields is the number field sieve (NFS) [9, 34, 35]. D. Weber has implemented this algorithm [37] and successfully computed discrete logarithms in a number of very large finite fields. Recently, he and T. Denny solved McCurley's discrete logarithm challenge, a discrete logarithm problem in a finite prime field for a 426-bit prime [38].

Let $\mathcal{O}_\Delta$ be any quadratic order. The best available algorithm for computing discrete logarithms in $Cl(\Delta)$ uses a generalization of the self-initializing quadratic sieve factoring algorithm [17]. The main idea behind this algorithm is as follows. First, compute the structure of $Cl(\Delta)$ as a direct product of cyclic subgroups,

$$Cl(\Delta) \simeq \bigotimes_{i=1}^{l} C(m_i),$$

together with generators $\mathfrak{g}_i$ of each cyclic subgroup (order of $[\mathfrak{g}_i]$ in $Cl(\Delta)$ is m_i). Then compute the representations

$$\mathfrak{a} \sim \prod_{i=1}^{l} \mathfrak{g}_i^{a_i}, \quad \mathfrak{b} \sim \prod_{i=1}^{l} \mathfrak{g}_i^{b_i}$$

of $\mathfrak{a}$ and $\mathfrak{b}$ over the generators. If we can find x satisfying

$$\prod_{i=1}^{l} \mathfrak{g}_i^{a_i} \sim \prod_{i=1}^{l} \mathfrak{g}_i^{x b_i},$$

then x is the discrete logarithm of $\mathfrak{a}$ to the base $\mathfrak{b}$. The integer x can be found by solving the system of simultaneous congruences

$$a_i \equiv x b_i \pmod{m_i}, \quad 1 \leq i \leq l, \tag{3}$$

using the generalized Chinese remainder theorem. If (3) cannot be solved, the given discrete logarithm problem has no solution.

The first problem which must be solved in order to implement this method is to compute the structure of $Cl(\Delta)$. We use the method described in [16] (Algorithm 4.3). Suppose we have computed a factor base $FB = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_k\}$ consisting of invertible prime ideals such that the equivalence classes of some subset of FB generates $Cl(\Delta)$. For $\vec{v} \in \mathbb{Z}^k$ we define

$$FB^{\vec{v}} = \prod_{i=1}^{k} \mathfrak{p}_i^{v_i}$$

where $\mathfrak{p}_i \in FB$. We call $\vec{v}$ a *relation* if $FB^{\vec{v}} \sim \mathcal{O}_\Delta$, i.e., the ideal given by $FB^{\vec{v}}$ is principal. A generating system $L = \{\vec{v}_1, \ldots, \vec{v}_n\}$ of the *relation lattice*

$$\Lambda = \{\vec{v} \in \mathbb{Z}^k \mid FB^{\vec{v}} \sim \mathcal{O}_\Delta\} \tag{4}$$

is then produced, which is the kernel of the homomorphism

$$\mathbb{Z}^k \to Cl(\Delta), \quad \vec{v} \to FB^{\vec{v}} \ . \tag{5}$$

Since the equivalence classes of the ideals of FB generate the class group, it follows that the homomorphism (5) is surjective, and we have

$$Cl(\Delta) \simeq \mathbb{Z}^k/\Lambda \ .$$

This implies that Λ is a k-dimensional lattice and its determinant is equal to $h(\Delta)$. Also, the *relation matrix* $A = (\vec{v}_1^T, \ldots, \vec{v}_n^T)$, the matrix formed by taking the relations $\vec{v}_i$ as columns, has rank k. The diagonal elements which are greater than 1 in S, the Smith normal form of A, are precisely the elementary divisors of $Cl(\Delta)$.

The major difference between this approach and that of earlier subexponential algorithms is in the way the generating system of the relation lattice is produced. The solution employed by earlier algorithms is to attempt to factor randomly produced ideals over the factor base. We replace this step by a sieve-based strategy similar to that used in the self-initializing quadratic sieve factoring algorithm [1]. We refer the interested reader to [16] for more details.

Once the structure of $Cl(\Delta)$ is computed, we have to compute representations of $\mathfrak{a}$ and $\mathfrak{b}$ over a system of generators of $Cl(\Delta)$. As shown in [17], the main work involved is essentially computing a single relation corresponding to $\mathfrak{a}$ and $\mathfrak{b}$. Compared to the time required to compute $Cl(\Delta)$ the time required to find these two extra relations is negligible. Hence, solving any instance of the discrete logarithm problem is relatively easy once we have computed $Cl(\Delta)$. See [17] for more details and computational results.

5. The DLP for arbitrary $Cl(\Delta_f)$

In this section we generalize the result from [15]. We show that given the conductor f and its prime factorization one can reduce the DLP in an arbitrary $Cl(\Delta_f)$ to the DLP in various smaller groups. More precisely, we first show that the computation of discrete logarithms in $Cl(\Delta_f)$ can be reduced to the computation of discrete logarithms in the class group $Cl(\Delta_1)$ of the maximal order and the computation of discrete logarithms in $\mathrm{Ker}(\phi_{Cl}^{-1})$. Furthermore, we show that the latter problem boils down to the computation of discrete logarithms in a small number of finite fields.

It should be noted that our method here is in essence a special case of the more general methods employed by Cohen et al. to compute discrete logarithms in ray class groups [5]. The class group of a non-maximal order in any number field, not only degree 2, can be viewed as a ray class group of the maximal order, where the modulus is simply an integer, the conductor of the non-maximal order. Our exposition here is a reformulation of these results in terms of the simpler, special case of non-maximal orders using the language of [15]. In addition, we prove that the reduction of the DLP in $Cl(\Delta_f)$ to computing discrete logarithm computations in $Cl(\Delta_1)$ and a small number of finite fields is of polynomial complexity.

We start with an algorithm which reduces the DLP in $Cl(\Delta_f)$ to the DLP in $Cl(\Delta_1)$ and $\mathrm{Ker}(\phi_{Cl}^{-1})$. Since the map $\psi : (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^* \rightarrow \mathrm{Ker}(\phi_{Cl}^{-1})$ given in Proposition 3 induces the isomorphism $\mathrm{Ker}(\phi_{Cl}^{-1}) \cong (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*/Im((\mathbb{Z}/f\mathbb{Z})^*)$, we will reduce the latter DLP to computations in $(\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*$. Thus, our algorithm makes use of the following two methods:

- $\mathsf{DLPinCl}(\mathfrak{G}, \mathfrak{A})$
  Accepts two reduced $\mathcal{O}_{\Delta_1}$-ideals $\mathfrak{G}, \mathfrak{A}$ as input and returns $x \in \mathbb{Z}$ with $0 \leq x < h(\Delta_1)$ such that $\mathfrak{G}^x \sim \mathfrak{A}$, or $x = -1$ if no such x exists.

- $\mathsf{DLPinKerphi}(\gamma, \alpha, |\mathrm{Ker}(\phi_{Cl}^{-1})|)$
  Accepts two generator representations γ, α of classes in $\mathrm{Ker}(\phi_{Cl}^{-1})$ such that $[\gamma], [\alpha] \in (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*$ as input and returns $x \in \mathbb{Z}$ with $0 \leq x < |\mathrm{Ker}(\phi_{Cl}^{-1})|$ such that $\psi([\gamma])^x = \psi([\alpha])$ in $\mathrm{Ker}(\phi_{Cl}^{-1})$, or $x = -1$ if no such x exists.

Furthermore, we assume that $h(\Delta_1)$ is known. This is no practical restriction, since the best currently known algorithm [16], as sketched in Section 4, needs to compute $h(\Delta_1)$ and the group structure of $Cl(\Delta_1)$ before the actual DL-computation starts. Secondly, if there were any other algorithm $\mathsf{DLPinCl}$ with the above properties, then one could compute $h(\Delta_1)$ as follows:

1. Use [16, Algorithm 3.2, p.33] to compute $h^* \in \mathbb{R}$, where $\frac{h^*}{2} < h(\Delta_1) < h^*$. This algorithm runs in polynomial time assuming the Extended Riemann Hypothesis (ERH).

2. Compute an arbitrary $\mathcal{O}_{\Delta_1}$-Ideal $\mathfrak{G} \neq \mathcal{O}_{\Delta_1}$. Set $x' = \lceil h^* \rceil$ and compute $x \leftarrow \mathsf{DLPinCl}(\mathfrak{G}, \mathfrak{G}^{x'})$. Then

$$h(\Delta_1) = \begin{cases} \frac{x'}{2}, & \text{if } x = 0 \\ x' - x, & \text{otherwise} \end{cases}$$

Proof. We have $x' \equiv x \pmod{h(\Delta_1)}$ and hence $x' - x = kh(\Delta_1)$ for some $k \in \mathbb{Z}$. We will derive bounds for k to show that only $k = 1$, or $k = 2$ if $x = 0$, is possible.

Assume $x = 0$. Then $x' = kh(\Delta_1)$ and $x' = \lceil h^* \rceil \geq h^* > h(\Delta_1)$ implies that $k > 1$. On the other hand, we have $x' = \lceil h^* \rceil < h^* + 1 < 2h(\Delta_1) + 1$ and hence $kh(\Delta_1) < 2h(\Delta_1) + 1$. Therefore we have $k < 2 + \frac{1}{h(\Delta_1)} \leq 3$, which implies $k = 2$.

Now assume $x > 0$. From $x' = \lceil h^* \rceil \geq h^* > h(\Delta_1)$ and $x < h(\Delta_1)$ it follows that $x' - x = kh(\Delta_1) > 0$, which implies $k \geq 1$. Furthermore we have

$$\begin{aligned} k &= \frac{x' - x}{h(\Delta_1)} = \frac{\lceil h^* \rceil - x}{h(\Delta_1)} < \frac{h^* + 1 - x}{h(\Delta_1)} \\ &< \frac{2h(\Delta_1) + 1 - x}{h(\Delta_1)} = 2 + \frac{1 - x}{h(\Delta_1)} \leq 2, \end{aligned}$$

which shows $k = 1$. □

Thus, assuming ERH, it is possible to reduce the computation of $h(\Delta_1)$ to DLPinCl in polynomial time and our assumption of the prior knowledge of $h(\Delta_1)$ is not a restriction if we assume that we can compute discrete logarithms in $Cl(\Delta_1)$.

Now we present our algorithm which reduces the DLP in $Cl(\Delta_f)$ to the DLP in $Cl(\Delta_1)$ and $\mathrm{Ker}(\phi_{Cl}^{-1})$.

Proof (correctness of ReduceDLP). Since the conductor f is known, one can compute $\mathfrak{G} = \phi_{Cl}^{-1}(\mathfrak{g}), \mathfrak{A} = \phi_{Cl}^{-1}(\mathfrak{a}) \in Cl(\Delta_1)$ and the discrete logarithm x_1 using DLPinCL. If $x_1 = -1$, then there is no discrete logarithm in $Cl(\Delta_1)$. Since $\phi_{Cl}^{-1} : Cl(\Delta_f) \to Cl(\Delta_1)$ is a surjective homomorphism, this would imply that the DLP in $Cl(\Delta_f)$ has no solution either, and we return -1 in this case. Otherwise, we have $x \equiv x_1 \pmod{h(\Delta_1)}$, i.e.,

$$x = c \cdot h(\Delta_1) + x_1, \tag{6}$$

for some $c \in \mathbb{Z}$. We assume that $h(\Delta_1)$ is known, so it remains to show how to compute c such that $0 \leq x < h(\Delta_f) = h(\Delta_1)|\mathrm{Ker}(\phi_{Cl}^{-1})|$. Since $0 \leq x_1 < h(\Delta_1)$, we see that $0 \leq c < |\mathrm{Ker}(\phi_{Cl}^{-1})|$.

Since $\mathfrak{G}^{x_1} \sim \mathfrak{A}$, and hence $\mathfrak{A}/\mathfrak{G}^{x_1} \sim \mathcal{O}_{\Delta_1}$, we have $\varphi(\mathfrak{A}/\mathfrak{G}^{x_1}) \sim \mathfrak{a}/\mathfrak{g}^{x_1} \in \mathrm{Ker}(\phi_{Cl}^{-1})$ and one may use Std2Gen (Algorithm 1) to compute a generator representation α of the class of $\varphi(\alpha\mathcal{O}_{\Delta_1}) \sim \mathfrak{a}/\mathfrak{g}^{x_1}$. In a similar fashion, we have $\mathfrak{g}^{h(\Delta_1)} \in \mathrm{Ker}(\phi_{Cl}^{-1})$ and one may compute a generator representation γ such that $\varphi(\gamma\mathcal{O}_{\Delta_1}) \sim \mathfrak{g}^{h(\Delta_1)}$. Now we solve

Algorithm 2 ReduceDLP

Input: Two reduced $\mathcal{O}_{\Delta_f}$-ideals $\mathfrak{g}, \mathfrak{a}$, the conductor f, the class number $h(\Delta_1)$, and the order of the kernel $|\mathrm{Ker}(\phi_{Cl}^{-1})| = \frac{f}{[\mathcal{O}_{\Delta_1}^*:\mathcal{O}_{\Delta_f}^*]} \prod_{p\,|\,f}\left(1-\frac{(\Delta/p)}{p}\right)$

Output: The discrete logarithm x, such that $\mathfrak{g}^x \sim \mathfrak{a}$, with $0 \leq x < h(\Delta_f)$, or $x = -1$, if no such x exists.

```
{Compute DL in Cl(Δ_1)}
𝔊 ←GoToMaxOrder(𝔤, f)
𝔄 ←GoToMaxOrder(𝔞, f)
x_1 ←DLPinCl(𝔊, 𝔄)
if x_1 = −1 then
  return(−1)
end if
{Compute DL in (O_{Δ_1}/f O_{Δ_1})^*}
α ←Std2Gen(𝔞/𝔤^{x_1}, f)
γ ←Std2Gen(𝔤^{h(Δ_1)}, f)
c ←DLPinKerphi(γ, α, |Ker(φ_{Cl}^{−1})|)
if c = −1 then
  return(−1)
end if
{Combine partial results to get DL in Cl(Δf^2)}
x ← c · h(Δ_1) + x_1
return(x)
```

the DLP in $\mathrm{Ker}(\phi_{Cl}^{-1})$, i.e., we compute c, such that $\psi([\gamma])^c = \psi([\alpha])$, with $0 \leq c < |\mathrm{Ker}(\phi_{Cl}^{-1})|$.

If such a c does not exist, then there is no solution to (6) for the DLP in $Cl(\Delta f^2)$ and we return -1. Assume now that such a c exists. Then we have $\psi([\gamma])^c = \psi([\alpha])$, or equivalently $\varphi(\gamma^c\mathcal{O}_{\Delta_1}) \sim \varphi(\alpha\mathcal{O}_{\Delta_1})$. Therefore

$$\mathfrak{g}^{h(\Delta_1)\cdot c} \sim \mathfrak{a}/\mathfrak{g}^{x_1} \sim \mathfrak{g}^{x-x_1} \tag{7}$$

and by equating exponents we obtain $x \equiv c \cdot h(\Delta_1) + x_1 \pmod{h(\Delta_f)}$. □

PROPOSITION 5. *Given the conductor f, the class number $h(\Delta_1)$ and the order of the kernel $|\mathrm{Ker}(\phi_{Cl}^{-1})|$ one can reduce the DLP in $Cl(\Delta_f)$ in $O(\log(|\Delta_f|)^3)$ bit-operations to the DLP in $Cl(\Delta_1)$ and $\mathrm{Ker}(\phi_{Cl}^{-1})$.*

Proof. GoToMaxOrder and Std2Gen both need $O(\log(|\Delta_f|)^2)$ bit-operations. Thus the dominating operations are the exponentiations in $Cl(\Delta_f)$. Since ideal multiplication and reduction in quadratic orders both have quadratic run-time [2], the result follows. □

Thus, in order to compute discrete logarithms in $Cl(\Delta_f)$, we need efficient algorithms for computing discrete logarithms in $Cl(\Delta_1)$ and $\mathrm{Ker}(\phi_{Cl}^{-1})$. The subexponential algorithm outlined in Section 4 is the most efficient algorithm known for computing discrete logarithms in $Cl(\Delta_1)$. We now consider the DLP in $\mathrm{Ker}(\phi_{Cl}^{-1}) \cong (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*/Im((\mathbb{Z}/f\mathbb{Z})^*)$ more closely.

By the Chinese Remainder Theorem (see, for example, [20, p.11]), the DLP in $(\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*/Im((\mathbb{Z}/f\mathbb{Z})^*)$ boils down to DLPs in $(\mathcal{O}_{\Delta_1}/p_i^{e_i}\mathcal{O}_{\Delta_1})^*/Im((\mathbb{Z}/p_i^{e_i}\mathbb{Z})^*)$ for prime powers $p_i^{e_i}$, where $f = \prod p_i^{e_i}$. Furthermore, this problem can be efficiently reduced to the prime case $(\mathcal{O}_{\Delta_1}/p_i\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_{p_i}^*)$. We give an algorithm (ReducePe2P) for this reduction, assuming that the following algorithm is available:

- DLPinOmodpO(γ, α)
  Accepts two elements $\gamma, \alpha \in (\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$ as input and returns $x \in \mathbb{Z}$ with $0 \leq x < |(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_p^*)|$ such that $[\gamma]^x \equiv [\alpha]$ in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_p^*)$, or $x = -1$ if no such x exists.

Proof (correctness of ReducePe2P). Let $n_1' = |(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_p^*)| = (p-\left(\frac{\Delta_1}{p}\right)$. Then it is easy to show that $n = |(\mathcal{O}_{\Delta_1}/p^e\mathcal{O}_{\Delta_1})^*/Im((\mathbb{Z}/p^e\mathbb{Z})^*)| = p^{e-1}n_1'$. This shows the correctness of n_1 and n_2 such that $n = n_1 \cdot n_2$. Since $\gcd(n_1, n_2) = 1$, we can compute x modulo n_1 and n_2, and in the end combine the partial results $x_1 \equiv x \pmod{n_1}$ and $x_2 \equiv x \pmod{n_2}$ using the Chinese Remainder Theorem. The correctness of the **for**-loop follows from the presentation of the algorithm in in [29, Algorithm 3.63, p.108]. Instead of using the Baby-Step Giant-Step algorithm, we compute $l_i \pmod{|(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*|}$ using DLPinOmodpO. If the discrete logarithm x modulo n_1, or the discrete logarithm in a p-order subgroup during the computation of x_2 does not exist, then the entire DLP is unsolvable and we return -1. □

Excluding the calls to DLPinOmodpO, the exponentiations are the dominating operations. Thus we obtain:

PROPOSITION 6. *The DLP in $(\mathcal{O}_{\Delta_1}/p^e\mathcal{O}_{\Delta_1})^*/Im((\mathbb{Z}/p^e\mathbb{Z})^*)$ can be reduced in $O(e\cdot(\log p^e)^3)$ bit-operations to $2e$ DL-computations in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_p^*)$.*

Algorithm 3 ReducePe2P

Input: Two elements $\gamma, \alpha \in (\mathcal{O}_{\Delta_1}/p^e\mathcal{O}_{\Delta_1})^*$.
Output: The discrete logarithm x such that $[\gamma]^x \equiv [\alpha]$ in $(\mathcal{O}_{\Delta_1}/p^e\mathcal{O}_{\Delta_1})^* / Im((\mathbb{Z}/p^e\mathbb{Z})^*)|$ with $0 \leq x < |(\mathcal{O}_{\Delta_1}/p^e\mathcal{O}_{\Delta_1})^* / Im((\mathbb{Z}/p^e\mathbb{Z})^*)|$, or $x = -1$ if no such x exists.

{Initialize $n_1, n_2 = p^k$ such that $n = n_1 n_2 = |(\mathcal{O}_{\Delta_1}/p^e\mathcal{O}_{\Delta_1})^*|$ and $\gcd(n_1, n_2) = 1$}
if $\left(\frac{\Delta_1}{p}\right) = 0$ **then**
  $n_1 \leftarrow 1$
  $k \leftarrow e$
else
  $n_1 \leftarrow \left(p - \left(\frac{\Delta}{p}\right)\right)$
  $k \leftarrow e - 1$
end if
$n_2 \leftarrow p^k$
$n \leftarrow n_1 \cdot n_2$
{Compute $x_1 \equiv x \pmod{n_1}$}
$x_1 \leftarrow$DLPinOmodpO(γ, α)
if $x_1 = -1$ **then**
  return(-1)
end if
$x_1 \leftarrow x_1 \pmod{n_1}$
{Compute $x_2 \equiv x \pmod{n_2}$, where $x_2 = l_0 + l_1 p + \cdots + l_{k-1} p^{k-1}$ and $0 \leq l_i < p$}
$\beta \leftarrow \mathcal{O}_{\Delta_1}$
$\overline{\gamma} \leftarrow \gamma^{n/p}$
$l_{-1} \leftarrow 0$
for $i = 0$ **to** $k - 1$ **do**
  $\beta \leftarrow \beta \gamma^{l_{i-1} p^{i-1}} \pmod{p^e \mathcal{O}_{\Delta_1}}$
  $\overline{\alpha} \leftarrow \left(\alpha \beta^{-1}\right)^{n/p^{i+1}} \pmod{p^e \mathcal{O}_{\Delta_1}}$
  $l_i \leftarrow$DLPinOmodpO$(\overline{\gamma}, \overline{\alpha})$
  if $l_i = -1$ **then**
    return(-1)
  end if
  $l_i \leftarrow l_i \pmod{p}$ {, where $0 \leq l_i < p$}
end for
$x_2 \leftarrow l_0 + l_1 p + l_2 p^2 + \cdots + l_{k-1} p^{k-1}$
Compute x using the CRT, such that $0 \leq x < n$, $x \equiv x_1 \pmod{n_1}$ and $x \equiv x_2 \pmod{n_2}$
return(x)

COROLLARY 1. *If $e = O((\log p)^\alpha)$ for some $\alpha = O(1)$, then the DLP in $(\mathcal{O}_{\Delta_1}/p^e\mathcal{O}_{\Delta_1})^*/Im((\mathbb{Z}/p^e\mathbb{Z})^*)$ can be reduced in polynomial time (in $\log p$) to the DLP in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_p^*)$.*

Using ReduceDLP and ReducePe2P allows us to reduce the DLP in $Cl(\Delta_f)$ to DLPs in $Cl(\Delta_1)$ and $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_p^*)$. As shown in [15, 14], $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$ is isomorphic to either $\mathbb{F}_p^* \times \mathbb{F}_p^*$ or $\mathbb{F}_{p^2}^*$, depending how p splits in $\mathcal{O}_{\Delta_1}$. This immediately leads to the central result of this section.

THEOREM 1. *If the prime factorization of the conductor $f = \prod_{i=1}^k p_i^{e_i}$ is known and $e_i = O((\log p_i)^\alpha)$ for some $\alpha = O(1)$ then one can reduce the discrete logarithm problem in $Cl(\Delta_f)$ in polynomial time (in $\log \Delta_f$) to the computation of logarithms in $Cl(\Delta_1)$ and the following groups $(1 \le i \le k)$:*

$$\begin{array}{l} \mathbb{F}_{p_i}^*, \text{ if } \left(\frac{\Delta_1}{p_i}\right) \in \{0, 1\} \\ \mathbb{F}_{p_i^2}^*, \text{ if } \left(\frac{\Delta_1}{p_i}\right) = -1 \ . \end{array}$$

Proof. If the conductor f and its prime factorization are known, then one can use ReduceDLP (Algorithm 2) to reduce the DLP in $Cl(\Delta_f)$ to the DLP in $Cl(\Delta_1)$ and $\mathrm{Ker}(\phi_{Cl}^{-1})$. By Proposition 5 this is possible in polynomial time in $\log \Delta_f$. By the Chinese Remainder Theorem (using the known factorization of f) the DLP in $\mathrm{Ker}(\phi_{Cl}^{-1}) \cong (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*/Im((\mathbb{Z}/f\mathbb{Z})^*)$ is nothing more than the DLP in groups of the form $(\mathcal{O}_{\Delta_1}/p_i^{e_i}\mathcal{O}_{\Delta_1})^*/Im((\mathbb{Z}/p_i^{e_i}\mathbb{Z})^*)$, which can, using ReducePe2P (Algorithm 3) and Corollary 1, be reduced in polynomial time (in $\log p_i$) to the DLP in $(\mathcal{O}_{\Delta_1}/p_i\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_{p_i}^*)$, because e_i is assumed to be polynomial in $\log p_i$.

It remains to show how one reduces the discrete logarithm problem in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_p^*)$ to discrete logarithm problems in $\mathbb{F}_p^*$ or $\mathbb{F}_{p^2}^*$. Suppose we have two representatives γ, α of classes in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$ for which we want to compute the discrete logarithm c such that $[\gamma]^c \equiv [\alpha]$ in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_p^*)$. In the inert case $(\Delta_1/p) = -1$, where $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^* \cong \mathbb{F}_{p^2}^*$, we have $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_p^*) \cong \mathbb{F}_{p^2}^*/Im(\mathbb{F}_p^*)$. It is well-known that there always exists a surjective homomorphism from $\mathbb{F}_{p^2}^*$ to $\mathbb{F}_{p^2}^*/Im(\mathbb{F}_p^*)$. Thus, we first solve the DLP $\gamma^{c'} \equiv \alpha \pmod{p\mathcal{O}_{\Delta_1}}$ by simply solving the corresponding DLP in $\mathbb{F}_{p^2}^*$. Taking $c \equiv c' \bmod (p+1)$ yields the required solution to the DLP $[\gamma]^c \equiv [\alpha]$ in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_p^*)$.

We now restrict our attention to the split case $(\Delta_1/p) = 1$, where we have $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^* \cong \mathbb{F}_p^* \times \mathbb{F}_p^*$. The element $\gamma = (x_1, y_1)$ maps to $(x_1 \bmod p, y_1 \bmod p) \in \mathbb{F}_p^* \times \mathbb{F}_p^*$ and similarly $\alpha = (x_2, y_2)$ maps to $(x_2 \bmod p, y_2 \bmod p)$. The DLP in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_p^*)$ becomes

$$(x_1, y_1)^c \equiv l(x_2, y_2) \quad (\text{in } \mathbb{F}_p^* \times \mathbb{F}_p^*)$$

which in turn yields the simultaneous DLP's

$$x_1^c \equiv l x_2 \pmod p, \quad y_1^c \equiv l y_2 \pmod p \ .$$

Since these two DLP's must be solved for the same c and l, we can combine them and obtain the single DLP in $\mathbb{F}_p^*$

$$\left(\frac{x_1}{y_1}\right)^c \equiv \left(\frac{x_2}{y_2}\right) \pmod p$$

from which we can find the desired value of c.

As noted in [11], this simple strategy can be used to improve the general maps from [15, 14]; it is shown that in this case there not only exists a surjective homomorphism $\mathbb{F}_p^* \times \mathbb{F}_p^* \to \mathrm{Ker}(\phi_{Cl}^{-1})$, but even an efficiently computable isomorphism $\mathbb{F}_p^* \cong \mathrm{Ker}(\phi_{Cl}^{-1})$. □

Note that the central result of [15] now is nothing more than an immediate corollary. The proof of Theorem 1 also describes an algorithm for computing discrete logarithms in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_p^*)$ (DLPinOmodpO).

5.1. Example

We illustrate the reduction of discrete logarithm computations in $Cl(\Delta_f)$ via a small example. Suppose $\Delta_1 = -1019$, $f = 23$, and $\Delta_f = \Delta_1 f^2 = -539051$. In this case, both $Cl(\Delta_f)$ and $Cl(\Delta_1)$ are cyclic with $h(\Delta_1) = 13$ and $h(\Delta_f) = h(\Delta_1)(23-1) = 286$. The equivalence class represented by the reduced ideal

$$\mathfrak{g} = 15\mathbb{Z} + \frac{-7+\sqrt{-539051}}{2}\mathbb{Z} = (15, -7)$$

generates $Cl(\Delta_f)$.

Suppose we wish to compute the discrete logarithm of $[\mathfrak{a}]$ with respect to the base $[\mathfrak{g}]$ in $Cl(\Delta_f)$, where

$$\mathfrak{a} = 11\mathbb{Z} + \frac{9+\sqrt{-539051}}{2}\mathbb{Z} = (11, 9) \ .$$

That is, we want to find x such that $\mathfrak{g}^x \sim \mathfrak{a}$. Since $\mathfrak{g}$ generates $Cl(\Delta_f)$, we know that such an x exists. Following ReduceDLP (Algorithm 2), we first compute $[\mathfrak{G}] = [\phi_{Cl}^{-1}(\mathfrak{g})]$ and $[\mathfrak{A}] = [\phi_{Cl}^{-1}(\mathfrak{a})]$, and solve the discrete logarithm problem

$$\mathfrak{G}^{x_1} \sim \mathfrak{A}$$

in $Cl(\Delta_1)$. We have $\mathfrak{G} = 15\mathbb{Z} + \frac{1+\sqrt{-1019}}{2}\mathbb{Z} = (15, 1)$, $\mathfrak{A} = (11, 9)$, and we easily compute $x_1 = 9$.

At this point we know that x has the form $x = c \cdot h(\Delta_1) + x_1 = 13c + 9$, and it remains to compute c. Again following ReduceDLP (Algorithm 2), we compute generator representations α, γ of $[\alpha], [\gamma] \in (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*$ such that $\psi([\alpha]) = [\mathfrak{a}/\mathfrak{g}^{x_1}]$ and $\psi([\gamma]) = [\mathfrak{g}^{h(\Delta_1)}]$. Following Std2Gen (Algorithm 1), we first compute

$$\mathfrak{b} \sim \mathfrak{a}/\mathfrak{g}^{x_1} \sim \mathfrak{a}/\mathfrak{g}^{9} = (311, 277)$$

and

$$\mathfrak{c} \sim \mathfrak{g}^{h(\Delta_1)} \sim \mathfrak{g}^{13} = (297, 295) \ .$$

To find α and γ we compute the principal ideals $\mathfrak{B} = \varphi^{-1}(\mathfrak{b})$ and $\mathfrak{C} = \varphi^{-1}(\mathfrak{c})$, and reduce them while simultaneously computing their modulo $f\mathcal{O}_{\Delta_1}$ reduced generators, which we take as α and γ. We obtain $\mathfrak{B} = (311, -15) = (\alpha)$ and $\mathfrak{C} = (297, -13) = (\gamma)$ where

$$\alpha = -8 + 1\omega, \quad \gamma = -7 + 1\omega$$

and $\omega = \frac{1+\sqrt{-1019}}{2}$.

To compute c, we need to solve the discrete logarithm problem

$$[\gamma]^c \equiv [\alpha] \quad (\text{in } \mathrm{Ker}(\phi_{Cl}^{-1}) \cong (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*/Im((\mathbb{Z}/f\mathbb{Z})^*)) \ .$$

For this example, we have $(\Delta_1/f) = (-1019/23) = 1$, and thus $(\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^* \simeq \mathbb{F}_{23}^* \times \mathbb{F}_{23}^*$ by [15, Lemma 8]. Since $\omega \equiv 14 \pmod{23}$ and $\overline{\omega} \equiv 10 \pmod{23}$, we obtain

$$\gamma \mapsto (-7 + 1\omega \bmod 23, -7 + 1\overline{\omega} \bmod 23) = (7, 3) \in \mathbb{F}_{23}^* \times \mathbb{F}_{23}^*$$

and

$$\alpha \mapsto (-8 + 1\omega \bmod 23, -8 + 1\overline{\omega} \bmod 23) = (6, 2) \in \mathbb{F}_{23}^* \times \mathbb{F}_{23}^* \ .$$

Since $\mathrm{Ker}(\phi_{Cl}^{-1}) \cong (\mathbb{F}_p^* \times \mathbb{F}_p^*)/Im(\mathbb{F}_p^*)$, we need to find c by solving the discrete logarithm problem $(7, 3)^c = l(6, 2)$ in $\mathbb{F}_{23}^* \times \mathbb{F}_{23}^*$ for every $l \in \mathbb{F}_{23}^*$. This yields

$$7^c \equiv 6l \pmod{23}, \quad 3^c \equiv 2l \pmod{23},$$

and we combine these two discrete logarithm problems to obtain one discrete logarithm problem in $\mathbb{F}_{23}^*$:

$$(7/3)^c \equiv (6/2) \pmod{23} \rightarrow 10^c \equiv 3 \pmod{23} \ .$$

Solving yields $c = 20$, and finally $x = 13 \cdot 20 + 9 = 269$. It is easy to verify that x is indeed the desired discrete logarithm: simply compute the reduced ideal $\mathfrak{g}^{269}$ and verify that it is equal to the reduced ideal $\mathfrak{a}$.

6. Towards practical non-interactive cryptosystems

In this section we apply (parts of) the result from Section 5 concerning the computation of discrete logarithms to set up a non-interactive cryptosystem.

Before we explain the proposed setup we recall some more preliminaries concerning imaginary quadratic class groups. Note that for fundamental discriminants, by the Cohen-Lenstra heuristics [6] the probability that the odd part of the class group is cyclic is approximately 0.977575. Thus, for a prime discriminant $-\Delta_1 \equiv 3 \pmod 4$ the probability that $Cl(\Delta_1)$ is cyclic is more than 0.97. Indeed, in practice it is no problem to find a fundamental discriminant Δ_1 such that the class group class group $Cl(\Delta_1)$ of the maximal order is cyclic. Furthermore, given such a maximal order, it is easy to find a prime conductor p such that $Cl(\Delta_p)$ is also cyclic.

PROPOSITION 7. *Let $q \equiv 3 \pmod 4$, $\Delta_1 = -q$ and let $Cl(\Delta_1)$ be cyclic with class number $h(\Delta_1)$. Furthermore let p be a prime such that*

$$\gcd\left(p - (\Delta_1/p), h(\Delta_1)\right) = 1 \ .$$

Then $Cl(\Delta_p)$ is cyclic.

Proof. By Proposition 2 we know that $\phi_{Cl}^{-1} : Cl(\Delta_p) \to Cl(\Delta_1)$ is a surjective homomorphism, and we have $Cl(\Delta_p) \simeq Cl(\Delta_p)/\mathrm{Ker}(\phi_{Cl}^{-1}) \times \mathrm{Ker}(\phi_{Cl}^{-1})$. Since $Cl(\Delta_p)/\mathrm{Ker}(\phi_{Cl}^{-1}) \simeq Cl(\Delta_1)$ is assumed to be cyclic, if we show that $\mathrm{Ker}(\phi_{Cl}^{-1})$ is cyclic, then by elementary group theory $Cl(\Delta_p)$, the direct product of two cyclic groups of relatively prime order (also by assumption), is also cyclic.

We know that $\mathrm{Ker}(\phi_{Cl}^{-1}) \cong (\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_p^*)$ (see Remark 1), and by [15, Lemma 8] $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$ is isomorphic to either $\mathbb{F}_p^* \times \mathbb{F}_p^*$ or $\mathbb{F}_{p^2}^*$. In the latter case, since $\mathbb{F}_{p^2}^*$ is cyclic, $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_p^*) \cong \mathbb{F}_{p^2}^*/Im(\mathbb{F}_p^*)$ must also be cyclic, since it is a factor group of a cyclic group.

Suppose now that $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^* \cong \mathbb{F}_p^* \times \mathbb{F}_p^*$. Then, we have $\mathrm{Ker}(\phi_{Cl}^{-1}) \cong (\mathbb{F}_p^* \times \mathbb{F}_p^*)/Im(\mathbb{F}_p^*)$ where

$$Im(\mathbb{F}_p^*) = \{(x, x) \mid x \in \mathbb{F}_p^*\}$$

It is easy to show that $(\mathbb{F}_p^* \times \mathbb{F}_p^*)/Im(\mathbb{F}_p^*) \cong \mathbb{F}_p^*$ under the map $x \mapsto (x, 1)Im(\mathbb{F}_p^*)$. $\square$

Thus, it is possible to set up a non-interactive scheme in the spirit of Maurer and Yacobi in a *cyclic* group $Cl(\Delta_p)$, where the embedding of some (arbitrarily large) identity ID_B into a group element is straightforward. One has only to take the largest prime $p_B \leq ID_B$ which satisfies $(\Delta_p/p_B) = 1$, compute the prime ideal $\mathfrak{p}_B$ lying over p_B, and reduce this ideal. The computation of the discrete logarithm can be performed in $Cl(\Delta_1)$ and the finite field $\mathbb{F}_p^*$ or $\mathbb{F}_{p^2}^*$, depending on (Δ_1/p), using the reduction described in Section 5 by anyone who knows the factorization of Δ_p.

Before we explain our system setup we list the crucial properties:

REQUIRED PROPERTIES:

1. The discrete logarithm problem (DLP) in $Cl(\Delta_p)$ *without* knowing the factorization of $\Delta_p = \Delta_1 p^2$ is infeasible. To determine bounds for Δ_1 and p, we make use of the heuristic model from [12], which is a refinement of Lenstra and Verheul's approach [21], since it also takes into account the asymptotically vanishing $o(1)$-part in subexponential algorithms. We will now derive bounds for the parameters such that an attacker would need to spend about $90,000$ MIPS years to break the system. This approximately amounts to a ten-fold higher workload than the recent factorization of RSA155 and hence corresponds to the very minimum requirements. The estimates in [12, Table 3] state that Δ_p should have at least $576, 667, 423$ bits to prevent factoring Δ_p with the GNFS, factoring Δ_p with ECM and computing discrete logarithms in $Cl(\Delta_p)$ with the SIQS-analogue [16], respectively.

   1.1 Δ_p is large enough that using the subexponential algorithm from [17] to directly compute discrete logarithms in $Cl(\Delta_p)$ is infeasible. $\Delta_p > 2^{423}$ implies an expected workload of more than $90,000$ MIPS years.

   1.2 Δ_p cannot be factored to reduce the DLP to DLPs in $Cl(\Delta_1)$ and $\mathbb{F}_p^*$ (or $\mathbb{F}_{p^2}^*$).

      1.2.1 Δ_p is large enough so that the Number Field Sieve would need more than $90,000$ MIPS years. This yields $\Delta_p > 2^{576}$.

      1.2.2 Δ_1 and p are large enough that it would take more than $90,000$ MIPS years to find them with the Elliptic Curve Method. This implies $\Delta_1, p > 2^{222}$.

2. Δ_1, p must be small enough to enable the KGC to compute discrete logarithms in $Cl(\Delta_1)$ and $\mathbb{F}_p^*$ using subexponential algorithms. $\Delta_1, p < 2^{300}$ seems to be feasible.

3. $Cl(\Delta_p)$ must be cyclic.

It is easy to see that the following setup satisfies *all* above requirements.

SYSTEM SETUP:

1. The KGC randomly chooses a prime $q \equiv 3 \pmod 4$, $q > 2^{260}$, sets $\Delta_1 = -q$ and computes $h(\Delta_1)$ and the group structure of $Cl(\Delta_1)$ with the algorithm from [16]. The Cohen-Lenstra heuristics [6] suggest that $Cl(\Delta_1)$ is cyclic with probability > 0.97. If $Cl(\Delta_1)$ is not cyclic, the KGC selects another prime q until it is cyclic.

2. The KGC chooses a prime $p > 2^{260}$ with $(\Delta_1/p) = 1$ and $\gcd(p - 1, h(\Delta_1) = 1$ such that the SNFS can be applied as in [38], and computes $\Delta_p = \Delta_1 p^2$. The gcd condition ensures that $Cl(\Delta_p)$ is cyclic.

3. The KGC computes a generator $\mathfrak{g}$ of $Cl(\Delta_p)$ and publishes it together with Δ_p.

Given a generator $\mathfrak{G}$ of $Cl(\Delta_1)$, which the KGC can easily obtain during the computation of $Cl(\Delta_1)$ [16, Algorithm 6.1], it is also easy in practice to find a generator $\mathfrak{g}$ of $Cl(\Delta_p)$ with the additional property that $\phi_{Cl}^{-1}(\mathfrak{g}) = \mathfrak{G}$. The KGC repeatedly selects random values of $\alpha \in \mathcal{O}_{\Delta_1}$ and takes the first $\mathfrak{g} = \phi(\alpha\mathfrak{G})$ such that $\mathfrak{g}^{h(\Delta_p)/d_i} \not\sim \mathcal{O}_{\Delta_p}$ for any positive divisor d_i of $h(\Delta_p)$. Although $h(\Delta_p)$ is approximately as large as $\sqrt{|\Delta_p|}$, in practice it has sufficiently many small factors that this condition can be verified with high probability.

USER REGISTRATION:

1. Bob requests the public key $\mathfrak{b}$ corresponding to his identity ID_B at the KGC.

2. The KGC verifies Bob's identity, for example, using a passport, and starts with the key generation.

3. The KGC computes the 128-bit hash $id = h(ID_B)$ using, for example, MD5 [33], of Bob's identity and embeds id into a group element of $Cl(\Delta_p)$ by taking the largest prime $p_B \leq id$, for which $(\Delta_p/p_B) = 1$ and computing the prime ideal $\mathfrak{b} = p_B\mathbb{Z} + \frac{b_B+\sqrt{\Delta_p}}{2}$, where b_B is the uniquely determined square root of $\Delta_p \bmod 4p_B$ with $0 \leq b_B \leq p_B$. Note that $\mathfrak{b}$ is already reduced, since $\sqrt{|\Delta_p|} > 2^{128} > p_B$. If the KGC recognizes that $\mathfrak{b}$ is already assigned to

another user it will ask Bob to choose another identity, for example, his postal address.

4. Finally, the KGC computes the discrete logarithm b such that $\mathfrak{g}^b \sim \mathfrak{b}$ using the secret knowledge of the conductor p and the reduction procedure described in the Section 5, and returns b to Bob.

As soon as all users are registered this way the KGC can destroy the factorization of Δ_p and cease to exist. The users can obtain any other user's *authentic* public key simply by hashing that user's identity and computing the largest prime ideal whose norm is less than the hash value. Each user has a public/private key-pair $(\mathfrak{a}, a)$ with $\mathfrak{a} \sim \mathfrak{g}^a$, so discrete logarithm-based protocols such as Diffie-Hellman or ElGamal can be directly applied in the class group $Cl(\Delta_p)$.

7. Practical experience

7.1. Example 1

As an example of setting up our system, we chose two primes q and p as described above:

$$\begin{aligned} q &= 30571674960498834085812920457916453747019461644031395 30/ \\ &\quad 79206249473499510535 30183 \\ &\quad (265 \text{ bits}) \\ p &= 21659790402940284734184104434358329064134399492163023 68/ \\ &\quad 39089285875389553863294923 \\ &\quad (267 \text{ bits}) \ . \end{aligned}$$

The KGC then takes

$$\begin{aligned} \Delta_1 &= -q \\ \Delta_p &= -qp^2 \text{ (798 bits)} \end{aligned}$$

as its maximal and non-maximal orders, respectively. Using a parallel version of the algorithm from [16] implemented in LiDIA [22] and PVM [8], it computes the structure of $Cl(\Delta_1)$ and a generator $\mathfrak{G}$. In this case $Cl(\Delta_1)$ is cyclic of order

$$h(\Delta_1) = 222447236471778012687215545272162414791 7 \ ,$$

and the ideal

$$\mathfrak{G} = (2, 1)$$

is a generator. Using a cluster of 16 Pentium-III/550 processors running LINUX, this computation took 2.27 days. The equivalence class represented by

$$
\begin{aligned}
\mathfrak{g} = (&12361476730058687124611460064973145361811372868012143 4/\\
&566216149445237284331225769882789615386139520933013131 8/\\
&666286517,\\
&393901110452399357327033752553095846044305881393475826 9/\\
&932606961090153671496518119850327947844291685752047120 1/\\
&0732255)
\end{aligned}
$$

generates $Cl(\Delta_p)$.

The parallelization of the class group algorithm from [16] is fairly straightforward. Since the relation generation stage of the algorithm is based closely on the SIQS factoring algorithm, the well-known parallelization techniques of that algorithm can be applied almost directly. In addition, a large portion of the linear algebra can be done in parallel, and in the end, if the number of processors is increased by a factor of n, we expect to achieve a speed-up of almost n. Details will be given in a forthcoming paper.

Now, suppose the users Alice ($ID_A =$ huehnlein@secunet.de), Bob ($ID_B =$ mjjacobs@cacr.math.uwaterloo.ca), and Carl ($ID_C =$ weber-@mfh-iserlohn.de) wish to register. Alice's public key, $\mathfrak{a}$, is computed by finding p_A, the largest prime less than the 128-bit MD5 hash value of ID_A with $(\Delta_p/p_A) = 1$, and taking the corresponding prime ideal in $\mathcal{O}_{\Delta_p}$. We obtain

$$
\begin{aligned}
\mathrm{MD5}(ID_A) &= 201149154589345561189246215978625621230,\\
p_A &= 201149154589345561189246215978625621213\\
\mathfrak{a} &= (201149154589345561189246215978625621213,\\
&\quad 98688405189933472976874523210781312655)\ .
\end{aligned}
$$

We find Bob and Carl's public keys in the same manner:

$$
\begin{aligned}
\mathrm{MD5}(ID_B) &= 185069019259970008578381740973744250599,\\
p_B &= 185069019259970008578381740973744250567\\
\mathfrak{b} &= (185069019259970008578381740973744250567,\\
&\quad 180969848314739057306605962939756582495)\\
\mathrm{MD5}(ID_C) &= 282020054827252238756640548852860917797,\\
p_C &= 282020054827252238756640548852860917779\\
\mathfrak{c} &= (282020054827252238756640548852860917779,\\
&\quad 233511465074997733088279729809348692979)\ .
\end{aligned}
$$

Note that anyone can compute these public keys given only Alice, Bob, and Carl's email addresses and the public system information Δ_p.

Next, the KGC computes Alice, Bob, and Carl's private keys, i.e. the discrete logarithms a, b, and c such that $\mathfrak{g}^a \sim \mathfrak{a}$, $\mathfrak{g}^b \sim \mathfrak{b}$, and $\mathfrak{g}^c \sim fc$ in $\mathcal{O}_{\Delta_p}$. The KGC knows p, the conductor of $\mathcal{O}_{\Delta_p}$, so it can use the method described in Section 5 to compute these discrete logarithms. Using the algorithm from [17] to compute the discrete logarithms in $Cl(\Delta_1)$ and that from [38] to compute the discrete logarithms in $\mathbb{F}_p^*$, we obtain

$$
\begin{aligned}
a &= 1108228913334518377434054522555907589914018044575075203/\\
&\quad 8524348119082017895936710359600058901396679232272945862/\\
&\quad 1035164570\\
b &= 3003480562979666120080073706524077910795497960511740471/\\
&\quad 6337520361425644457779075116232216932978464156827037738/\\
&\quad 9543408800\\
c &= 2745078930329044874097630185924164819719438358865808856/\\
&\quad 3627735994430126325251998111722295801030917594158201110/\\
&\quad 8170580376\ .
\end{aligned}
$$

Since the information used to compute $Cl(\Delta_1)$ and $\mathfrak{G}$ has been kept by the KGC, the computation of the discrete logarithms in $Cl(\Delta_1)$ is very fast in comparison to the initial setup of the system. In this case the discrete logarithm computations in $Cl(\Delta_1)$ for a, b, and c each took only about 3.10 minutes each using the Pentium cluster. As in the computation of $Cl(\Delta_1)$, increasing the number of machines by a factor of n will yield a speed-up of almost n, so these computations are completely feasible for a KGC with even rather modest amounts of computing resources.

Using a single 500 Mhz Pentium III, the computation of the discrete logarithms in $\mathbb{F}_p^*$ each took about 2.3 hours. However, most of the computation of the discrete logarithms in $\mathbb{F}_p^*$ is also trivially parallelizable, resulting in a linear speed-up for all stages except the linear algebra.

7.2. Example 2

Due to recent advances in the efficiency of the elliptic curve factoring method, the parameters used in the previous example are on the borderline of security. Computing the structure of the class group, and hence discrete logarithms, in quadratic orders with arbitrary discriminants of more than 265 bits is quite difficult. Fortunately, it is possible to choose the prime q in such a way that this computation is much easier that that for an arbitrary discriminant. As pointed out in [16], if the discriminant Δ_1 is such that $(\Delta_1/l) = 1$ for many small primes l, then computing the class group is significantly easier in practice. Such special discriminants can be generated easily using numerical sieving devices such as the MSSU [24].

Thus, for our second example, we chose two primes q and p as described above:

$$
\begin{aligned}
q &= 488863540868851242629626061867149873587159640479347634 9/\\
&\quad 0607621218132307601660728438392515591\\
&\quad (305 \text{ bits})\\
p &= 3047718323340697663928401918879192361689531023359999999/\\
&\quad 9999999999999999999999999999999999987\\
&\quad (304 \text{ bits}) \ .
\end{aligned}
$$

In this case, the prime q was found using the MSSU and the method describe in [18], and has the additional property that $(-q/l) = 1$ for all primes $l < 389$. The KGC then takes

$$
\begin{aligned}
\Delta_1 &= -q\\
\Delta_p &= -qp^2 \text{ (913 bits)}
\end{aligned}
$$

as its maximal and non-maximal orders, respectively. Again, using a parallel version of the algorithm from [16] it computes the structure of $Cl(\Delta_1)$ and a generator $\mathfrak{G}$. In this case $Cl(\Delta_1)$ is cyclic of order

$$h(\Delta_1) = 2486756768703544308000520498386078077471807140 3 \ ,$$

and the ideal

$$\mathfrak{G} = (2, 1)$$

is a generator. Using the Pentium cluster, this computation took 2.87 days. The equivalence class represented by

$$
\begin{aligned}
\mathfrak{g} = \ &(7749732490542104870872621685800289463053977629304598 52/\\
&3969468387224151041578132740295101542448943011314827539/\\
&0152822738457063050752368204,\\
&2884013129991658377584730954547970251314084045462303210/\\
&8878544708721954266311440661112152691706666261864358048/\\
&7342298715627422256969331 93)
\end{aligned}
$$

generates $Cl(\Delta_p)$.

Now, suppose as before the users Alice (ID_A = huehnlein@secunet.de), Bob (ID_B = mjjacobs@cacr.math.uwaterloo.ca), and Carl (ID_C =

weber@mfh-iserlohn.de) wish to register. We obtain the following public keys:

$$\begin{aligned}
\mathfrak{a} &= (20114915458934556118924621597862562094 9,\\
&\quad 79174889249695020620380553564943557169),\\
\mathfrak{b} &= (18506901925997000857838174097374425059 1,\\
&\quad 92632418736137979645746252873831476417),\\
\mathfrak{c} &= (28202005482725223875664054885286091773 3,\\
&\quad 48832595588063748011907744520095764189)\quad .
\end{aligned}$$

The corresponding private keys, i.e., the discrete logarithms a, b, and c such that $\mathfrak{g}^a \sim \mathfrak{a}$, $\mathfrak{g}^b \sim \mathfrak{b}$, and $\mathfrak{g}^c \sim \mathfrak{c}$ in $\mathcal{O}_{\Delta_p}$, are

$$\begin{aligned}
a &= 355806955044151450525410241623491252595466407068278585 6/\\
&\quad 052827721874906785587212418938702193022238827992845003 0/\\
&\quad 259459108394080934750004067 6\\
b &= 554397239648033388164765969805197169213931097026569694 4/\\
&\quad 058464733626518107589008270325277153875484456368533227 5/\\
&\quad 688771445998851815164607675 9\\
c &= 624675808839721216073641957870656409843965704939995699 5/\\
&\quad 816117269685595562701314841727462406336871859943241313 0/\\
&\quad 278501917860454254748981955\quad .
\end{aligned}$$

The discrete logarithm computations in $Cl(\Delta_1)$ for a, b, and c each took only about 3.30 minutes on the Pentium cluster, and those in $\mathbb{F}_p^*$ each took about 14 hours each on a single 500 Mhz Pentium III. Thus, although the initial start-up costs are higher, it is still feasible to set up our non-interactive system with sufficiently large parameters to provide reasonable security.

Acknowledgements

The authors greatfully acknowledge the assistance of Renate Scheidler for her careful proof-reading of several versions of the manuscript, and in particular for correcting the proofs of Theorem 1 and Proposition 7.

References

1. W.R. Alford and C. Pomerance, *Implementing the self-initializing quadratic sieve on a distributed network*, Proceedings of International Conference "Number Theoretic and Algebraic Methods in Computer Science" (Moscow, 1993) (A.J. van der Poorten, I. Shparlinski, and H.G Zimmer, eds.), World Scientific, 1995, pp. 163–174.
2. I. Biehl and J. Buchmann, *An analysis of the reduction algorithm for binary quadratic forms*, Voronoi's Impact on Modern Science (Kyiv, Ukriaine) (P. Engel and H. Syta, eds.), vol. 1, Institute of Mathematics of National Academy of Sciences, 1999.
3. Z.I. Borevich and I.R. Shafarevich, *Number theory*, Academic Press, New York, 1966.
4. H. Cohen, *A course in computational algebraic number theory*, Springer-Verlag, Berlin, 1993.
5. H. Cohen, F. Diaz y Diaz, and M. Olivier, *Computing ray class groups, conductors, and discriminants*, Math. Comp. **67** (1998), no. 222, 773–795.
6. H. Cohen and H.W. Lenstra, Jr., *Heuristics on class groups of number fields*, Number Theory, Lecture notes in Math., vol. 1068, Springer-Verlag, New York, 1983, pp. 33–62.
7. D.A. Cox, *Primes of the form* $x^2 + ny^2$, John Wiley & Sons, New York, 1989.
8. A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine - a user's guide and tutorial for networked parallel computing*, MIT Press, Cambridge, Mass., 1994.
9. D. Gordon, *Discrete logarithms using the number field sieve*, Siam J. Discrete Math. **6** (1993), 124–138.
10. D. Hühnlein, *Efficient implementation of cryptosystems based on non-maximal imaginary quadratic orders*, Selected Areas in Cryptography - SAC'99, Lecture Notes in Computer Science, vol. 1758, 1999, pp. 150–167.
11. D. Hühnlein, *Faster generation of NICE-Schnorr signatures*, In preparation, preprint via http://www.informatik.tu-darmstadt.de/TI/Veroeffentlichung/TR/Welcome.html#2000, 2000.
12. D. Hühnlein, *Quadratic orders for NESSIE – overview and parameter sizes of three public key families*, Submitted to ISSE 2000, preprint via http://www.informatik.tu-darmstadt.de/TI/Veroeffentlichung/TR/Welcome.html#2000, 2000.
13. D. Hühnlein, M.J. Jacobson, Jr., S. Paulus, and T. Takagi, *A cryptosystem based on non-maximal imaginary quadratic orders with fast decryption*, Advances in Cryptology - EUROCRYPT '98, Lecture Notes in Computer Science, vol. 1403, 1998, pp. 294–307.
14. D. Hühnlein and J. Merkle, *An efficient NICE-Schnorr-type signature scheme*, To appear at PKC2000 and in Springer LNCS, preprint via http://www.informatik.tu-darmstadt.de/TI/Veroeffentlichung/TR/Welcome.html#1999, 2000.
15. D. Hühnlein and T. Takagi, *Reducing logarithms in totally non-maximal imaginary quadratic orders to logarithms in finite fields*, Advances in Cryptology - ASIACRYPT '99, Lecture Notes in Computer Science, 1999.
16. M.J. Jacobson, Jr., *Subexponential class group computation in quadratic orders*, Ph.D. thesis, Technische Universität Darmstadt, Darmstadt, Germany, 1999.
17. M.J. Jacobson, Jr., *Computing discrete logarithms in quadratic orders*, Journal of Cryptology **13** (2000), 473–492.

18. M.J. Jacobson, Jr. and H.C. Williams, *The size of the fundamental solutions of consecutive Pell equations*, Exp. Math. **9** (2000), no. 4, 631–640.
19. D. Kügler, *Eine Aufwandsanalyse für identitätsbasierte Kryptosysteme*, Master's thesis, Technische Universität Darmstadt, Darmstadt, Germany, 1998, (in German), via http://www.informatik.tu-darmstadt.de/TI/Veroeffentlichung.
20. S. Lang, *Algebraic number theory*, 2nd ed., Springer, Berlin, 1991, ISBN 3-540-94225-4.
21. A.K. Lenstra and E. Verheul, *Selecting cryptographic key sizes*, Proceedings of Public Key Cryptography 2000, Lecture Notes in Computer Science, vol. 1751, 2000, pp. 446–465.
22. The LiDIA Group, *LiDIA: a* `C++` *library for computational number theory*, Software, Technische Universität Darmstadt, Germany, 1997, See http://www.informatik.tu-darmstadt.de/TI/LiDIA.
23. C.H. Lim and P.J. Lee, *Modified Maurer-Yacobi's scheme and its applications*, Proceedings of Auscrypt'92, Lecture Notes in Computer Science, 1992, pp. 308–323.
24. R.F. Lukes, C.D. Patterson, and H.C. Williams, *Numerical sieving devices: Their history and some applications*, Nieuw Archief voor Wiskunde (4) **13** (1995), 113–139.
25. M. Maurer and D. Kügler, *A note on the weakness of the Maurer-Yacobi squaring method*, Tech. report, Department of Computer Science, Technical University of Darmstadt, Darmstadt, Germany, 1999, To appear.
26. U. Maurer and Y. Yacobi, *Non-interactive public-key cryptography*, Advances in Cryptology - EUROCRYPT'91, Lecture Notes in Computer Science, vol. 547, 1991, pp. 498–507.
27. U. Maurer and Y. Yacobi, *A remark on a non-interactive public-key distribution system*, Advances in Cryptology - EUROCRYPT'92, Lecture Notes in Computer Science, vol. 658, 1993, pp. 458–460.
28. U. Maurer and Y. Yacobi, *A non-interactive public-key distribution system*, Design Codes and Cryptography **9** (1996), 305–316.
29. A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of applied cryptography*, Series on discrete mathematics and its applications, CRC Press, Boca Raton, 1996, ISBN 0-8493-8523-7.
30. J. Neukirch, *Algebraische zahlentheorie*, Springer, Berlin, 1992.
31. T. Okamoto and S. Uchiyama, *Security of an identity based cryptosystem and related reductions*, Advances in Cryptology - EUROCRYPT'98, Lecture Notes in Computer Science, vol. 1403, 1998, pp. 546–560.
32. J.M. Pollard, *Theorems on factorization and primality testing*, Proc. Cambridge Philos. Society **76** (1974), 521–528.
33. R. Rivest, *The MD5 message-digest algorithm*, 1992, RFC1321, Internet Activities Board, Internet Engineering Task Force.
34. O. Schirokauer, *Discrete logarithms and local units*, Theory and applications of numbers without large prime factors (R.C. Vaughan, ed.), Philos. Trans. Roy. Soc. London Ser. A, vol. 345, The Royal Society, London, 1993, pp. 409–423.
35. O. Schirokauer, *Using number fields to compute logarithms in finite fields*, Math. Comp. **69** (2000), 1267–1283.
36. A. Shamir, *Identity based cryptosystems and signature schemes*, Advances in Cryptology - CRYPTO '84, Lecture Notes in Computer Science, vol. 196, 1985, pp. 47–53.

37. D. Weber, *Computing discrete logarithms with the number field sieve*, Algorithmic Number Theory - ANTS-II (Université Bordeaux I, Talence, France), Lecture Notes in Computer Science, vol. 1122, Springer-Verlag, Berlin, 1996.
38. D. Weber and T. Denny, *The solution of McCurley's discrete log challenge*, Advances in Cryptology - CRYPTO '98, Lecture Notes in Computer Science, vol. 1462, 1998, pp. 56–60.

Address for Offprints:
Department of Computer Science
University of Manitoba
Winnipeg, Manitoba, Canada, R3T 2N2