Designs, Codes and Cryptography, 30, 281–299, 2003

Towards Practical Non-Interactive Public-Key Cryptosystems Using Non-Maximal Imaginary Quadratic Orders

DETLEF HÜHNLEIN detlif.huehnlein@secunet.de
secunet Security Networks AG, Sudetenstrasse 16, D-96247 Michelau, Germany

MICHAEL J. JACOBSON, JR. jacobs@cpsc.ucalgary.ca
Department of Computer Science, University of Calgary, 2500 University Drive NW, Calgary, Alberta, Canada, T2N 1N4

DAMIAN WEBER weber@informatik.fh-trier.de
Fachhochschule Trier, Schneidershof, D-54293 Trier, Germany

Communicated by: P. Wild

Received April 9, 2001; Revised April 22, 2002; Accepted May 21, 2002

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 $(\mathbb{Z}/n\mathbb{Z})^*$ [25,27] 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 specific case of the more general algorithm used in the setting of ray class groups [5]. We prove—for arbitrary non-maximal orders—that this 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

AMS Classification: 94A60, 11Y40

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.* 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 [33]. 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 [25]. They proposed setting up a discrete logarithm based system in $G = (\mathbb{Z}/n\mathbb{Z})^*$, where $n = p_1, \ldots, p_r$, 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 describing an algorithm that 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 a generic group-theoretic method employed to compute discrete logarithms in ray class groups [5]. We prove that this 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 Maurer and Yacobi [26] and Maurer and Kügler [24] demonstrate it seems to be no trivial 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 analog of the self-initializing quadratic sieve (SIQS)

*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 $\mathfrak{b}$.

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 [35].

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. Section 4 contains the discrete logarithm algorithm for arbitrary non-maximal imaginary quadratic orders. In Section 5 we present our new non-interactive public-key cryptosystem, followed by computational examples in Section 6.

2. Previous Proposals of Non-Interactive Cryptosystems

Although the paradigm of identity based cryptography was already introduced by Shamir in 1984 [33], it seems that Maurer and Yacobi [25] 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 $n = p_1, \ldots, p_r$ 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 [25] proposed two embeddings which solve this problem. Unfortunately, one method is insecure (allows a single user to factor n) and the second is inefficient.

Kügler [19] 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.

Currently, the best proposal for identity based cryptography, due to Boneh and Franklin [2], uses the Weil pairing of an elliptic curve. In addition to having efficient and secure private key generation, their scheme has chosen ciphertext security in the random oracle model assuming a variant of the computational Diffie-Hellman problem. Our cryptosystem is certainly not as attractive as that of Boneh and Franklin [2]. However, as it is based on an unrelated mathematical problem, it should be considered as a viable alternative should Boneh and Franklin's method be broken.

3. Background and Notation for Non-Maximal Imaginary Quadratic Orders

The basic notions of imaginary quadratic number fields can be found in Borevich and Shafarevich [3] and Cohen [4]. For a more comprehensive treatment of the relationship between maximal and non-maximal orders we refer to Cox [7], Hühnlein et al. [13], Hühnlein and Takagi [15], and Neukirch [29].

3.1. Maximal Imaginary Quadratic Orders

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

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

where

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

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 $\Delta \equiv 0 \ (\text{mod}\ 4)$) 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 Cohen [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 Cox [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 `GoToMaxOrder` $(\mathfrak{a}, f)$ to compute φ^{-1} and `GoToNonMaxOrder` $(\mathfrak{A}, f)$ to compute φ respectively can be found in Hühnlein et al. [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 $\mathfrak{A}, \mathfrak{B} \in \mathcal{I}_{\Delta_f}(f)$ 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:

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

(see Neukirch [29, 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 Cox [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 Cox [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, Jacobson [16, Algorithm 2.6, p.16]).

Algorithm 1. `Std2Gen`

```
Input: The standard representation $(a, b)$ of a reduced $\mathcal{O}_{\Delta_f}$-ideal $\mathfrak{a} = a\mathbb{Z} +$
  $(b + \sqrt{\Delta_f}/2\mathbb{Z})$ representing a class in $\mathrm{Ker}(\phi_{Cl}^{-1})$, and the conductor $f$.
Output: A generator representation $(x, y)$ of the class $[\mathfrak{a}] \in \mathrm{Ker}(\phi_{Cl}^{-1})$
  $(a, b) \leftarrow$ GoToMaxOrder $(\mathfrak{a}, f)$
  $(\mathfrak{G}, \gamma) \leftarrow$ Reduce $(a, b)$, where $\gamma = (x + y\sqrt{\Delta_f})/2$
  if $\mathfrak{G} \not\sim \mathcal{O}_{\Delta_1}$ then
    return ("Error! $\mathfrak{a} \notin \mathrm{Ker}(\phi_{Cl}^{-1})$!")
  end if
  if $\Delta_1 \equiv 0 \pmod 4$ then
    $\bar{x} \leftarrow x/2 \pmod f$
    $\bar{y} \leftarrow y/2 \pmod f$
  else
    $\bar{x} \leftarrow (x - y)/2 \pmod f$
    $\bar{y} \leftarrow y \pmod f$
  end if
  return $((\bar{x}, \bar{y}))$
```

Proof (correctness of `Std2Gen`*).* The first step in the routine `GoToMaxOrder` [13] is to compute an ideal $\mathfrak{a}' \sim \mathfrak{a}$ with $\gcd(\mathcal{N}(\mathfrak{a}'), f) = 1$). 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 Biehl and Buchmann [1] that the reduction, including the computation of γ, also takes $O((\log|\Delta_f|)^2)$ bit-operations. ■

4. The DLP for Arbitrary $Cl(\Delta_f)$

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 the least positive $a \in \mathbb{Z}$ such that $\mathfrak{g}^a = \mathfrak{a}$, or show that no such a exists.

In this section 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 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. [5] to compute discrete logarithms in ray class groups. 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. The following theorem, presented in a generic group-theoretic setting, encapsulates the general method employed.

THEOREM 1. *Let G, H be finite abelian groups and $\Phi : G \mapsto H$ a homomorphism. Then the DLP in G can be reduced to one discrete logarithm computation in* $\mathrm{Ker}(\Phi)$ *and two discrete logarithm computations in H via two applications of* Φ, $O(\log|H|)$ *group operations in G and* $O((\log|G|)^2)$ *bit-operations.*

Proof. Let $\mathfrak{g}, \mathfrak{a} \in G$ and suppose we want to compute the discrete logarithm x of $\mathfrak{a}$ to the base $\mathfrak{g}$. We first compute $\mathfrak{G} = \Phi(\mathfrak{g})$, $\mathfrak{A} = \Phi(\mathfrak{a})$, and solve the corresponding DLP in H, i.e., find the smallest positive integer x_1 such that $\mathfrak{G}^{x_1} = \mathfrak{A}$. If no such x_1 exists, then x does not exist because Φ is a homomorphism.

Let $\mathfrak{a} = \mathfrak{g}^{x_1}\mathfrak{h}$. Then the discrete logarithm x exists if and only if $\mathfrak{h} = \mathfrak{a}\mathfrak{g}^{-1} \in \langle\mathfrak{g}\rangle \cap \mathrm{Ker}(\Phi)$. The smallest power $\mathfrak{g}^u$ such that $\mathfrak{g}^u \in \mathrm{Ker}(\Phi)$ is a generator of the cyclic group $\langle\mathfrak{g}\rangle \cap \mathrm{Ker}(\Phi)$. Because Φ is a homomorphism, u is the order of $\Phi(\mathfrak{g}) = \mathfrak{G} \in H$. We compute u by solving a second DLP in H. Compute the smallest positive integer u' such that $\mathfrak{G}^{u'} = \mathfrak{G}^{-1}$; then $u = u' + 1$ is the order of $\mathfrak{G} \in H$.

Finally, we compute the discrete logarithm v of $\mathfrak{h}$ with respect to $\mathfrak{g}^u$ in the group $\mathrm{Ker}(\Phi)$, so that $\mathfrak{h} = \mathfrak{g}^{uv}$. Thus, we have $\mathfrak{a} = \mathfrak{g}^{x_1}\mathfrak{g}^{uv}$, and $x = x_1 + uv$. Because u, v, and x_1 are the smallest positive solutions to the corresponding discrete logarithm problems, x is the smallest positive solution to the original DLP.

In addition to the two applications of Φ and the three discrete logarithm computations, we require $O(\log|H|)$ group operations in G to compute $\mathfrak{h} = \mathfrak{a}\mathfrak{g}^{-x_1}$ and $\mathfrak{g}^u$ because x_1 and u are $O(|H|)$. Finally, because x is $O(|G|)$, we require $O(\log^2|G|)$ bit-operations to compute $x = x_1 + uv$. ■

If $|H|$ is known and $\mathfrak{g}$ generates G, as will be the case in our application, then we can simply set $u = |H|$ and avoid one discrete logarithm computation in H because $\Phi(\mathfrak{g})$ generates H (easy exercise). This yields Algorithm 2, in which we assume that the following subalgorithms are available:

- `ComputePhi`($\mathfrak{g}$)
  Accepts an element $\mathfrak{g} \in G$ as input and returns $\mathfrak{G} = \Phi(\mathfrak{g}) \in H$,
- `DLPinH`($\mathfrak{G}, \mathfrak{A}$)
  returns $x \in \mathbb{Z}$ with $0 \leq x < |\langle \mathfrak{G} \rangle|$ such that $\mathfrak{G}^x = \mathfrak{A}$ in H, or $x = -1$ if no such x exists,
- `DLPinKer`($\mathfrak{g}, \mathfrak{a}$)
  returns $x \in \mathbb{Z}$ with $0 \leq x < |\mathrm{Ker}(\Phi)|$ such that $\mathfrak{g}^x = \mathfrak{a}$ in $\mathrm{Ker}(\Phi)$, or $x = -1$ if no such x exists.

Algorithm 2. `ReduceDLP`

```
Input: Two elements $\mathfrak{g}, \mathfrak{a} \in G$ ($\mathfrak{g}$ generates $G$), $u = |H|$
Output: The discrete logarithm $x$, such that $\mathfrak{g}^x \sim \mathfrak{a}$, with $0 \leq x < |G|$, or $x = -1$, if no
  such $x$ exists.
  {Compute DL in $H$}
  $\mathfrak{G} \leftarrow$ ComputePhi($\mathfrak{g}$)
  $\mathfrak{A} \leftarrow$ ComputePhi($\mathfrak{a}$)
  $x_1 \leftarrow$ DLPinH($\mathfrak{G}, \mathfrak{A}$)
  if $x_1 = -1$ then
    return $(-1)$
  end if
  {Compute DL in $H$}
  $v \leftarrow$ DLPinKer ($\mathfrak{g}^u, \mathfrak{a}\mathfrak{g}^{-x_1}$)
  if $v = -1$ then
    return $(-1)$
  end if
  {Combine partial results to get DL in $G$}
  $x \leftarrow x_1 + uv$
  return($x$)
```

In our application, $G \leftarrow Cl(\Delta_f)$ (class group of the non-maximal order), $H \leftarrow Cl(\Delta_1)$ (class group of the maximal order), and $\Phi \leftarrow \phi_{Cl}^{-1}$. The subalgorithm `ComputePhi` is implemented by the algorithm `GoToMaxOrder` from Hühnlein et al. [13]. It remains to show how the subalgorithms `DLPinH` and `DLPinKer` are to be realized.

DLP in $Cl(\Delta)$

Because $H \leftarrow Cl(\Delta_1)$, the subalgorithm `DLPinH` must solve the DLP in the class group of a maximal order. 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]. This first step of this algorithm is to compute the structure of the class group; once this is complete, any DLP instances in that class group are relatively easy to solve. We refer the interested reader to Jacobson [17] for more details and computational results.

DLP in $\mathrm{Ker}(\phi_{Cl}^{-1})$

Because $\Phi \leftarrow \phi_{Cl}^{-1}$, the subalgorithm `DLPinKer` must solve the DLP in $\mathrm{Ker}(\phi_{Cl}^{-1})$. The map $\psi : (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^* \to \mathrm{Ker}(\phi_{Cl}^{-1})$ from Proposition 3 induces the isomorphism $\mathrm{Ker}(\phi_{Cl}^{-1}) \cong (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*/Im((\mathbb{Z}/f\mathbb{Z})^*)$, so we will reduce the computation of DLP's in $\mathrm{Ker}(\phi_{Cl}^{-1})$ to computations in $(\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*$. Our implementation of `DLPinKer` will accept 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 return $x \in \mathbb{Z}$ 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.

By the Chinese Remainder Theorem (see, for example, Lang [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 the groups $(\mathcal{O}_{\Delta_1}/p_i^{e_i}\mathcal{O}_{\Delta_1})^*/Im((\mathbb{Z}/p_i^{e_i}\mathbb{Z})^*)$ 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 $(\mathcal{O}_{\Delta_1}/p_i\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_{p_i}^*)$ using an analogous strategy to that in Menezes et al. [28, Algorithm 3.63, p. 108]. Excluding the discrete logarithm computations in $(\mathcal{O}_{\Delta_1}/p_i\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_{p_i}^*)$, exponentiations (polynomial time) are the dominating operations in the resulting algorithm. As shown in Hühnlein and Merkle [14] and Hühnlein and Takagi [15], $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$ is isomorphic to either $\mathbb{F}_p^* \times \mathbb{F}_p^*$ or $\mathbb{F}_{p^2}^*$, depending how p splits in $\mathcal{O}_{\Delta_1}$.

The best available algorithm for computing discrete logarithms in finite fields is the number field sieve (NFS) [9,31,32]. Weber has implemented this algorithm [34] and successfully computed discrete logarithms in a number of very large finite fields. Recently, he and Denny solved McCurley's discrete logarithm challenge, a discrete logarithm problem in a finite prime field for a 426-bit prime [35].

The Main Theorem

THEOREM 2. *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* $(1 \leq i \leq k)$:

$$\mathbb{F}_{p_i}^*, \quad \text{if} \left(\frac{\Delta_1}{p_i}\right) \in \{0, 1\}$$

$$\mathbb{F}_{p_i^2}^*, \quad \text{if} \left(\frac{\Delta_1}{p_i}\right) = -1.$$

Proof. If the conductor f and its prime factorization are known, then by Theorem 1 (using Algorithm 2) one can reduce the DLP in $Cl(\Delta_f)$ to the DLP in $Cl(\Delta_1)$ and $\mathrm{Ker}(\phi_{Cl}^{-1})$. By Theorem 1 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 be reduced in polynomial time (in $\log p_i$) to the DLP in $(\mathcal{O}_{\Delta_1}/p_i\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_{p_i}^*)$, as long as e_i is 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 v such that $[\gamma]^v \equiv [\alpha]$ 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 $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_p^*) \cong {}^{\sim}\mathbb{F}_{p^2}^*/Im(\mathbb{F}_p^*)$. 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 $\gamma^{v'} \equiv \alpha \pmod{p\mathcal{O}_{\Delta_1}}$ by simply solving the corresponding DLP in $\mathbb{F}_{p^2}^*$. Taking $v \equiv v' \mathrm{mod}(p+1)$ yields the required solution to the DLP $[\gamma]^v \equiv [\alpha]$ in the group $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_p^*)$.

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 $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/Im \times (\mathbb{F}_p^*)$ becomes

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

which in turn yields the simultaneous DLP's

$$x_1^v \equiv lx_2 \pmod{p}, \quad y_1^v \equiv ly_2 \pmod{p}.$$

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

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

from which we can find the desired value of v. ■

As noted in Hühnlein [11], this simple strategy can be used to improve the general maps from Hühnlein and Merkle [14] and Hühnlein and Takagi [15]; 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 Hühnlein and Takagi [15] now is nothing more than an immediate corollary. The proof of Theorem 2 also describes an algorithm for computing discrete logarithms in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/Im(\mathbb{F}_p^*)$.

4.1. Example

We illustrate the reduction of discrete logarithm computations in $Cl(\Delta_f)$ via a small example. Suppose $\Delta_1 = -1019$, $f = 23$, and $\Delta_f = \Delta_1 f^2 = -539\,051$. 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

$$\mathfrak{g} = 15\mathbb{Z} + \frac{-7+\sqrt{-539\,051}}{2}\mathbb{Z} = (15, -7),$$

generates $Cl(\Delta_f)$, so we can take $u = h(\Delta_1) = 13$.

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

$$\mathfrak{a} = 11\mathbb{Z} + \frac{9+\sqrt{-539\,051}}{2}\mathbb{Z} = (11, 9).$$

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

At this point we know that x has the form $x = x_1 + uv = 9 + 13v$, and it remains to compute v. 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 $\omega = 1 + \sqrt{-1019}/2$.

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

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

For this example, we have $(\Delta_1/f) = (-1019/23) = 1$, and thus the group

$(\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^* \simeq \mathbb{F}_{23}^* \times \mathbb{F}_{23}^*$ by Hühnlein and Takagi [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 v by solving the discrete logarithm problem $(7, 3)^v = l(6, 2)$ in $\mathbb{F}_{23}^* \times \mathbb{F}_{23}^*$ for every $l \in \mathbb{F}_{23}^*$. This yields

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

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

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

Solving yields $v = 20$, and finally $x = 9 + 13 \cdot 20 = 269$. 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 $\mathfrak{a}$.

5. Towards Practical Non-Interactive Cryptosystems

In this section we apply (parts of) the result from Section 4 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 5. *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*

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

Then $Cl(\Delta_p)$ is cyclic.

Proof. By Proposition 2 we know that $\phi_{Cl}^{-1} : Cl(\Delta_p) \rightarrow Cl(\Delta_1)$ 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 Hühnlein and Takagi [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 $Im(\mathbb{F}_p^*) = \{(x,x) | x \in \mathbb{F}_p^*\}$. It is easy to show that $(\mathbb{F}_p^* \times \mathbb{F}_p^*)/Im(\mathbb{F}_p^*) \cong \mathbb{F}_p^*$ under the map $(x,1)Im(\mathbb{F}_p^*) \mapsto x$. ■

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 4 by anyone who knows the factorization of Δ_p.

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

Required Properties:

1. The 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 Hühnlein [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 Hühnlein [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-analog [16], respectively.

   1.1. Δ_p is large enough that using the subexponential algorithm from Jacobson [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 Jacobson [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 Weber and Denny [35], 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 [30], 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 $\mathfrak{b} = p_B\mathbb{Z} + (b_B + \sqrt{\Delta_p})/2$, 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 4, 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 $Cl(\Delta_p)$.

6. Practical Experience

6.1. Example 1

As an example of setting up our system, we chose two primes q (265 bits) and p (267 bits) as described above, and set $\Delta_1 = -q$ and $\Delta_p = -qp^2$ (798 bits). Using a parallel version of the algorithm from Jacobson [16] implemented in LiDIA [22] and PVM [8], we computed the structure of $Cl(\Delta_1)$ and a generator $\mathfrak{g}$ of $Cl(\Delta_p)$ in 2.27 days on a cluster of 16 Pentium-III/550 processors running LINUX (36.32 on a single machine).

The parallelization of the class group algorithm from Jacobson [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.

To demonstrate the assignment of private keys by a KGC, we embedded three e-mail addresses into prime ideals of $\mathcal{O}_{\Delta_p}$ and computed their discrete logarithms with respect to $\mathfrak{g}$ using the method described in Section 4. The algorithm from Jacobson [17] was used to compute the discrete logarithms in $Cl(\Delta_1)$ and that from Weber and Denny [35] to compute the discrete logarithms in $\mathbb{F}_p^*$.

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 three discrete logarithm computations in $Cl(\Delta_1)$ 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.

6.2. 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 Jacobson [16], if the discriminant Δ_1 a quadratic residue modulo 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 [23].

Thus, for our second example, we chose two primes q (305 bits) and p (304 bits). In this case, the prime q was found using the MSSU and the method described in Jacobson and Williams [18], and has the additional property that $(-q/l) = 1$ for all primes $l < 389$. We then took $\Delta_1 = -q$ and $\Delta_p = -qp^2$ (913 bits), and using a parallel version of the algorithm from Jacobson [16], computed the structure of $Cl(\Delta_1)$ and a generator $\mathfrak{g}$ of $Cl(\Delta_p)$. Using the Pentium cluster, this computation took 2.87 days.

As before, we computed discrete logarithms of three prime ideals in $\mathcal{O}_{\Delta_p}$ with respect to $\mathfrak{g}$ using the method described in Section 4. The discrete logarithm computations in $Cl(\Delta_1)$ 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.

Acknowledgments

The authors gratefully 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 2 and Proposition 5.

References

1. 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).
2. D. Boneh and M. Franklin, *Identity based encryption from the Weil Pairing*, Advances in Cryptology – CRYPTO 2001, Lecture Notes in Computer Science, Vol. 2139 (2001) pp. 213–229.
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.*, Vol. 67, No. 222 (1998) pp. 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.*, Vol. 6 (1993) pp. 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, *Topics in Cryptology—CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001*, Lecture Notes in Computer Science, Vol. 2020 (2001) pp. 1–12.
12. D. Hühnlein, *Quadratic orders for NESSIE – overview and parameter sizes of three public key families*, Technical Report No. TI-3/00, TU-Darmstadt, via http://www.informatik.tu-darmstadt.de/TI/Welcome.html, 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, *Proceedings of PKC 2000*, Melbourne, Lecture Notes in Computer Science, Vol. 1751 (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*, Vol. 13 (2000) pp. 473–492.
18. M. J. Jacobson, Jr. and H. C. Williams, The size of the fundamental solutions of consecutive Pell equations, *Exp. Math.*, Vol. 9, No. 4 (2000) pp. 631–640.
19. D. Kügler, *Eine Aufwandsanalyze 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*, Second Edition, 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. R. F. Lukes, C. D. Patterson and H. C. Williams, Numerical sieving devices: Their history and some applications, *Nieuw Archief voor Wiskunde*, Vol. 13, No. 4 (1995) pp. 113–139.
24. 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.
25. 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.
26. 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.
27. U. Maurer and Y. Yacobi, A non-interactive public-key distribution system, *Design Codes and Cryptography*, Vol. 9 (1996) pp. 305–316.
28. 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.
29. J. Neukirch, *Algebraische zahlentheorie*, Springer, Berlin (1992).

30. R. Rivest, *The MD5 message-digest algorithm*, 1992, RFC1321, Internet Activities Board, Internet Engineering Task Force.
31. 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.
32. O. Schirokauer, Using number fields to compute logarithms in finite fields, *Math. Comp.*, Vol. 69 (2000) pp. 1267–1283.
33. A. Shamir, Identity based cryptosystems and signature schemes, *Advances in Cryptology—CRYPTO '84*, Lecture Notes in Computer Science, Vol. 196 (1985) pp. 47–53.
34. 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).
35. 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.