A Survey on IQ Cryptography

Johannes Buchmann Safuat Hamdy

March 21, 2001

$\mathbf{A}\mathbf{bstract}$

This paper gives a survey on cryptographic primitives based on class groups of imaginary quadratic orders (IQ cryptography, IQC). We present IQC versions of several well known cryptographic primitives, and we explain, why these primitives are secure if one assumes the hardness of the underlying problems. We give advice on the selection of the cryptographic parameters and show the impact of this advice on the efficiency of some IQ cryptosystems.

1 Introduction

The term IQ cryptography (IQC) refers to cryptography based on class groups of imaginary quadratic orders. IQC has been invented in 1988 [6]. Therefore, IQC is of about the same age as ECC (elliptic curve cryptography) [23, 18], yet IQC didn't get the same attention. What was most lacking was a comprehensive guide for using IQC. That is, there is still no formal document that describes how to select the cryptographic parameters and which cryptographic schemes to use. Advice on the selection of the cryptographic parameters for IQC has been given in [14]; a summary of some results of that paper has been included in this work. The aim of this paper is to present IQC versions of some well known cryptographic schemes, to describe the algorithms that can be used for the underlying arithmetic, and to discuss the performance of IQC schemes using those algorithms. Thus, this paper is another step towards a standardization of IQC.

Although class groups are ordinary finite abelian groups, some cryptographic schemes based on discrete logarithms can't be used with class groups in a straight forward way. The reason is that the order of class groups (or odd divisors thereof) can't be computed efficiently. However, some discrete logarithm based cryptographic schemes, for example any signature scheme of ElGamal type, require the knowledge of the group order. In order to use these schemes, it is necessary to modify them. In this paper we present such modifications for DSA. It also turns out that class groups are well suited for the Guillou-Quisquater signature scheme.

Finally, the performance of IQ cryptosystems has never been compared to the performance of established cryptosystems. Since the performance depends on the size of the cryptographic parameter, which in turn depends on the desired security level, the selection of the cryptographic parameter had to be investigated first. This has been done in [14]. In this paper first realistic benchmarks for IQ cryptosystems are presented, where the cryptographic parameter has been chosen of such a size that solving the discrete logarithm problem in the class group is about as hard as solving the integer factoring problem for integers of certain size. The result is that IQ cryptosystems appear to be practical. Since comparably little reserach has been spent on efficient IQ arithmetic, it is reasonable to expect significant improvements in this area in the future.

This paper is organized as follows: Sect. 2 recalls some relevant facts and notations from the theory of imaginary quadratic number fields. In Sect. 3 we discuss the computational problems on which IQ cryptosystems are based, and we discuss some properties of class groups that are relevant to cryptographic applications. In Sect. 4 we present IQ versions of some well known cryptographic schemes. Finally, in Sect. 5 we present benchmarks for an IQC variant of DSA and compare them with benchmarks of traditional cryptosystems.

2 Basic notation

We shall briefly recall some notations that we shall use in the sequel (see [16] or [4] for full details). Let Δ be a negative integer such that $\Delta \equiv 0, 1 \pmod{4}$. Then the ring $\mathcal{O}_{\Delta} = \mathbb{Z} + (\Delta + \sqrt{\Delta})/2\mathbb{Z}$ is an *imaginary quadratic order* of *discriminant* Δ . Its field of fractions is $\mathbb{Q}(\sqrt{\Delta})$. The discriminant Δ is called *fundamental* if $\Delta/4$ or Δ is square free for $\Delta \equiv 0 \pmod{4}$ or $\Delta \equiv 1 \pmod{4}$, respectively. (If Δ is fundamental, then \mathcal{O}_{Δ} is a *maximal* order.)

The fractional ideals of any imaginary quadratic order are of the form $q(a\mathbb{Z} + (b + \sqrt{\Delta})/2\mathbb{Z})$ with $q \in \mathbb{Q}$, $a, b \in \mathbb{Z}$, a > 0, $4a \mid b^2 - \Delta$, and gcd(a, b, c) = 1 where $c = (b^2 - \Delta)/(4a)$. Hence, they can be represented by triples (q, a, b). If q = 1, then the ideal is called *intregral*. Two ideals $\mathfrak{a}_1, \mathfrak{a}_2 \subseteq \mathcal{O}_\Delta$ are called *equivalent* if there is a non-zero number $\alpha \in \mathbb{Q}(\sqrt{\Delta})$ such that $\mathfrak{a}_2 = \alpha \mathfrak{a}_1$.

The set of equivalence classes forms an abelian group under ideal multiplication. This group is called *class group* and denoted by $Cl(\Delta)$. The class group is always finite. Its order is called *class number* and is denoted by $h(\Delta)$. The class number is not efficiently computable, if the discriminant is fundamental, but the even part of $h(\Delta)$ can be efficiently computed, if the prime factorization of Δ is known. For example, if Δ is a negative prime, then $h(\Delta)$ is always odd.

To compute with equivalence classes of class groups, one has to select representatives from each class. From the definition of equivalence it is obvious that any non-zero fractional ideal is equivalent to an integral one. Thus, each equivalence class can be represented by an integral ideal. Additionally, the product of two integral ideals is also an integral ideal. Therefore, we shall deal only with integral ideals, which we shall represent by pairs (a, b). Moreover, class groups of imaginary quadratic orders have the property that each equivalence class of ideals contain exactly one *reduced* integral ideal. An integral ideal (a, b) of a quadratic order \mathcal{O}_{Δ} is called reduced, if the following conditions are satisfied, where $c = (b^2 - \Delta)/(4a)$: 1. $a \leq c$, 2. $-a < b \leq a$, and 3. if a = c, then b > 0.

There are efficient algorithms to reduce integral ideals (i.e. finding an equivalent reduced ideal), see Sect. 5. Thus, we shall represent each equivalence class by a unique reduced integral ideal. Computing with reduced ideals is usually optimal in terms of efficiency because the bit sizes of reduced ideals are small. In particular, if (a, b) is a reduced ideal of \mathcal{O}_{Δ} , then $a \leq \sqrt{|\Delta|/3}$. Since $|b| \leq a$, the bit size of (a, b) is at most that of Δ . The multiplication of reduced ideals takes $O(\log^2 |\Delta|)$ bit operations. Since the product of reduced ideals is usually not reduced, we shall reduce any (intermediate) result. A careful analysis of the reduction algorithm presented in Sect. 5 shows that this algorithm takes also $O(\log^2 |\Delta|)$ bit operations, and therefore, a group operation (i.e. ideal multiplication with subsequent reduction) takes $O(\log^2 |\Delta|)$ bit operations.

3 Security of IQC

In this section we shall make some statements on the hardness of some computational problems for class groups and on the selection of IQC parameters.

3.1 Some computational problems

Let G be a finite abelian group. Then we define

Discrete logarithm problem (DLP): given $\alpha, \beta \in G$, find the smallest positive integer x such that $\beta = \alpha^x$ (or deceide that no such x exists).

Order problem (OP): given $\alpha \in G$, compute $|\langle \alpha \rangle|$.

Root problem (RP): given $\alpha \in G$ and an integer x > 1, compute β such that $\beta^x = \alpha$ (or decide that no such $\beta \in G$ exists).

Since $|\langle \alpha \rangle| = \mathsf{DLP}(\alpha, \mathbf{1}_G)$, we have $\mathsf{OP} \leq \mathsf{DLP}$. Conversely, the knowledge of the group order does apparently not help to compute discrete logarithms. For example, the group order of multiplicative

groups of any finite field is obviously known, yet the computation of discrete logarithms still appears to be intractable. Thus, it is unlikely that OP = DLP.

Finally, there are efficient methods to compute an *x*th root (or to decide that such a root does not exist), if we know the group order. Thus, $\mathsf{RP} \leq \mathsf{OP}$. It is an open question, whether $\mathsf{RP} = \mathsf{OP}$ or not, and it is also unknown, whether it is possible to compute roots efficiently without knowing the group order.

In the context of class groups of imaginary quadratic orders we denote these three problems by IQ-DLP, IQ-OP, and IQ-RP. It is known that $IQ-DLP \leq IFP^1$, so the complexity to solve the IFP is a lower bound for the complexity to solve the IQ-DLP.

IQ-DLP, IQ-OP, and IQ-RP appear to be hard problems. Despite the fact, that these problems don't appear to be equivalent (with respect to complexity theory), the best known algorithms to solve each of these problems are variants of each other with the same asymptotic running time. More precisely, there is no better method known to compute a solution to the IQ-RP than to compute a solution to the IQ-OP, for which in turn is no better method known than to compute a solution to an instance of the IQ-DLP.

The security of the IQ cryptosystems are based either on the IQ-DLP, or on the IQ-RP. Since instances of the IQ-RP are solved by invoking an algorithm for the IQ-DLP, we shall focus on the IQ-DLP. The following properties of class groups are of major interest for cryptographic applications:

• The class group is large, if the discriminant is large. It was known to Gauß that the average class number $\overline{h(\Delta)} = c_1 \sqrt{D}$ for all fundamental discriminants up to -D, where $c_1 \approx 0.46$. In fact, from the Brauer-Siegel Theorem [19] follows that

$$\sqrt{|\Delta|}^{1+\epsilon} \le h(\Delta) \le \sqrt{|\Delta|}^{1-\epsilon} \tag{1}$$

For any positive real ϵ . Moreover, if one assumes the Extended Riemann Hypothesis, then it is possible to show [21] that

$$\frac{1+o(1)}{c_2\ln\ln|\Delta|}\sqrt{|\Delta|} < h(\Delta) < (1+o(1))c_3\sqrt{|\Delta|}\ln\ln|\Delta|$$
(2)

where $c_2 = 12e^{\gamma}/\pi \approx 6.8$ and $c_3 = 2e^{\gamma}/\pi \approx 1.134$.

- The class group of a randomly chosen fundamental discriminant contains a very large cyclic subgroup with very high probability. This follows from [9, conjecture C5]
- The probability that the class number of a randomly chosen fundamental discriminant is very smooth, is negligible. This follows from [9, conjecture C2] and an additional assumption [7], see [14].

This shows, that class groups are suitable for cryptographic purposes. The above algorithms are called *generic*, for they work in any finite abelian group. As we shall see in the next subsection, there are faster algorithms known to compute discrete logarithms in class groups that are not generic.

3.2 Choosing the cryptographic parameter

Since the class group depends only on the discriminant, the discriminant is the main cryptographic parameter. In [14] all known strategies to compute discrete logarithms in class groups have been investigated. These are:

• Reductions to discrete logarithm computations in multiplicative groups of finite fields. Such reduction has been found for *totally non-maximal orders*. Conversely, there are no such reduction known for maximal orders, hence the discriminant should be chosen to be fundamental. The simplest way to achieve this is to select a (large) prime p and set $\Delta = -p$ or to select two (large) primes p and q and set $\Delta = -pq$ (such that, in both cases, $\Delta \equiv 1 \pmod{4}$).

¹Integer factorization problem

- A method similar to the (p-1)-factoring algorithm to compute the class number and the Pohlig-Hellman algorithm. This (p-1)-like method can be used to compute the class number, if it is very smooth (a similar algorithm has been used in an other factoring algorithm due to Schnorr and H.W. Lenstra [28]). Then, the Pohlig-Hellman algorithm can be used for discrete logarithm computations. The (p-1) algorithm has exponential running time in the size of the smoothness bound. In [14] it was shown that the probability that the class number of a random fundamental discriminant is very smooth is negligible. Hence, we expect the (p-1) algorithm to have exponential running time on average.
- Square-root algorithms, such as the Baby-Step-Giant-Step method or the ρ and λ algorithms (see [22, Chap. 3] for an overview and further references). The running time of these algorithms is exponential in the size of the class group, and from the previous section follows, that they are also exponential in the size of the discriminant
- Index-calculus algorithms. These algorithms have subexponential running time (and need subexponential space) in the size of the discriminant (see [17] or [8, Sect. 5.5]). Therefore, in order to protect IQ cryptosystems from attacks by index-calculus algorithms, much larger discriminants are required than for any square-root algorithm or the (p-1)-algorithm.

The fastest known algorithm to solve the IQ-DLP is a variant of the MPQS factoring algorithm (IQ-MPQS). It is asymptotically much slower than the GNFS, the fastest known algorithm to factor integers (see Figure 1). If one compares the expected running times of the GNFS and the IQ-MPQS, one gets that factoring 1024-bit integers requires about as much computational work as the computation of class groups with a 687-bit discriminant. Similar results are summarized in the Table 1.

4 IQC protocols

We shall now describe some protocols for IQC. Since class groups are ordinary finite abelian groups, we could use any scheme that is based on discrete logarithms. However, the class number (i.e. the order of a class group) is usually not efficiently computable. Thus, cryptographic schemes that require the knowledge of the group order (such as DSA or the Schnorr signature scheme) can't be used in a straight forward manner. But these protocols can modified in such a way that the knowledge of the group order is not required.

From the above introduction it is clear that protocols like the Diffie-Hellman key exchange or the ElGamal encryption scheme can be used in a straight forward manner with class groups.

4.1 The ElGamal encryption

The ElGamal encryption scheme could be used just as described in [11]. However, this would require a user to embed the plain text into a group element. In [26] probabilistic methods have been presented

Table	1:	Estimated	expected	computational	work	of the	GNFS	for	factoring	${\rm integers}$	and	$_{\mathrm{the}}$	IQ-
MPQS	5 for	computin	g discrete	logarithms in (class g	roups a	ligned						

magni	tude of			
n	$ \Delta $	expected no. of MIPS-years		
2^{768}	2^{540}	$4.99 imes10^7$		
2^{1024}	2^{687}	$6.01 imes10^{10}$		
2^{1536}	2^{958}	5.95×10^{15}		
2^{2048}	2^{1208}	$7.05 imes10^{19}$		
2^{3072}	2^{1665}	$2.65 imes 10^{26}$		
2^{4096}	2^{2084}	$5.87 imes10^{31}$		



Figure 1: Asymptotic expected running times for the GNFS and the IQ-MPQS. Here we assume the expected running times to be $L_n\left[\frac{1}{3}, \sqrt[3]{64/9}\right]$ for the GNFS and $L_{|\Delta|}\left[\frac{1}{2}, 1\right]$ for the IQ-MPQS. The details can be found in [14].

to do this efficiently, but neither of the described methods are very fast in practice. It is also not really necessary to embed a group elements. Instead, one can use the following modified ElGamal signature:

- **Key Generation:** A randomly selects a fundamental discriminant Δ of size according to the desired security level. Then A randomly selects $\gamma \in Cl(\Delta)$, $a \leq \sqrt{|\Delta|}$, and computes $\alpha = \gamma^a$. A's public key is (Δ, γ, α) , the private key is a.
- **Encryption:** To encrypt the plain text m, B randomly selects $k \leq \sqrt{|\Delta|}$ and computes $\kappa = \gamma^k$ and $\beta = \alpha^k$. Then the cipher text is $C = (\kappa, c)$, where $c = m \oplus f(\beta)$, f is a preimage and collision resistant hash function, and \oplus denotes bitwise xor-ing.

Decryption: A computes $\beta = \kappa^a$ and recovers m by computing $m = c \oplus f(\beta)$.

This scheme actually resembles the Diffie-Hellman key exchange, where A enters the protocol at the key generation stage, and B completes the protocol on encrypting the plain text. The same idea has been used for elliptic curves in [1]. This scheme is deterministic and very efficient.

Finally, observe that the Cramer-Shoup encryption scheme [10] can be used in a straight forward way with class groups. Moreover, the same modification we have proposed for the IQ version of the ElGamal encryption scheme above can be applied to the Cramer-Shoup encryption scheme.

4.2 Two DSA variants

As noted before, DSA [12] or the Schnorr [27] signature scheme (and similar signature schemes of ElGamal type) can't be used in a straight forward way with class groups. This is because the class number, i.e. the order of the class group can't be computed efficiently, at least if the discriminant is fundamental. But signing a message with a signature scheme of ElGamal type requires a reduction modulo the group order or a divisor thereof. We demonstrate this with a generalized version of DSA:

Key Generation: A randomly selects a group G, such that |G| has a 160-bit prime divisor q (the actual size of G depends on properties of G and the specific security requirements). Then A randomly selects $\gamma_0 \in G$ and computes

$$\gamma = \gamma_0^{|G|/q} \ . \tag{3}$$

(If $\gamma = 1_G$ A selects another γ_0 .) Finally, A randomly selects $a \leq 2^{160}$ and computes $\alpha = \gamma^a$. A's public key is (G, q, γ, α) , the private key is a.

Signature: To sign the message m, A randomly selects $k \leq 2^{160}$ and computes $\rho = \gamma^k$, an integer $r = f(\rho)$, and an integer

$$s = k^{-1} \left(f(m) + ar \right) \mod q \quad , \tag{4}$$

where f is a collision and preimage resistant hash function. Then the signature for m is S = (r, s), where f is a preimage and collision resistant hash function.

Verification: B checks that 0 < s < q. Then B computes $w = s^{-1} \mod q$, $u_1 = wf(m) \mod q$, $u_2 = wr \mod q$, and $v = f(\gamma^{u_1} \alpha^{u_2})$. Finally, B checks that v = r.

Without knowledge of |G| the computation of (3) and (4) is impossible.

In the context of class groups (i.e. $G = Cl(\Delta)$), (3) can be replaced by a random selection of $\gamma \in Cl(\Delta)$, because $Cl(\Delta)$ contains large cyclic subgroups with very high probability. Moreover, the probability that $h(\Delta)$ is very smooth is negligible. Hence, by group theoretic arguments, the probability that $\langle \gamma \rangle$ is large and $|\langle \gamma \rangle|$ has a large prime divisor is very high. It remains to answer the question: what can we do about (4)? There are at least two possible solutions:

- 1. Replace q by an integer that is not related to $h(\Delta)$. The main motivation for this is the fact that $s = x \mod q$ is equivalent to $s = x \ell q$ for some integer ℓ
- 2. Omit the modular reduction.

First, in order to simplify the matter, we modify the signature scheme such that the modular inversion of (4) disappears. To do this, we pick DSA variant EG II.3 from [15], in which (4) becomes

$$s = -af(m, \varrho) + k \mod q \quad , \tag{5}$$

with appropriate modification of the signature (which becomes $S = (\varrho, s)$) and the verification procedure. The resulting scheme is actually more similar to the Schnorr signature scheme than to DSA, but for technical reasons we shall call it DSA, too. In the following subsections we shall discuss the two solutions.

4.2.1 Replacing the modulus

This variant of DSA has been described in [3]. It is called RDSA. The security of the resulting protocol is based on the *root problem* (and *not* on the discrete logarithm problem), whence the name.

- **Key generation:** A randomly selects a fundamental discriminant Δ . Then A randomly selects $\gamma \in Cl(\Delta)$ and a 160-bit prime q. Then A randomly selects $a \leq 2^{160}$ and computes $\alpha = \gamma^a$. A's public key is $(\Delta, q, \gamma, \alpha)$, the private key is a.
- **Signature:** To sign the message m, A randomly selects $k \leq 2^{160}$ and computes $\varrho = \gamma^k$ and an integer $x = -af(m, \varrho) + k$. Then A divides x by q with remainder, i.e. A computes integers s and ℓ such that $x = q\ell + s$ and $0 \leq s < q$. Finally, A computes $\lambda = \gamma^{\ell}$. Then the signature for m is $S = (s, \varrho, \lambda)$.

Verification: B checks that $0 \le s < q$ and that $\gamma^s \alpha^{f(m,\varrho)} \lambda^q = \varrho$.

From this description it is obvious that an attacker, who can compute qth roots in $Cl(\Delta)$, can forge signatures for A: He simply randomly selects $\varrho \in Cl(\Delta)$ and $s \in \{0, \ldots, q-1\}$, then he sets $\lambda = (\varrho \gamma^{-s} \alpha^{-f(m,\varrho)})^{1/q}$. It is easily checked that (s, ϱ, λ) is a valid signature for m under A's public key.

In [3] the converse has been shown in the random oracle model [2], i.e. an attacker who can efficiently compute existentially forged signatures in a chosen message attack, can efficiently compute qth roots. Therefore, under the assumption of the hardness of the root problem, RDSA is secure against existential forgeries even in a chosen message attack.

4.2.2 Omitting the modular reduction

This variant of DSA is due to an idea in [25]. Unlike the DSA variant in the previous subsection, this variant is based on the discrete logarithm problem.

- Key generation: A randomly selects a fundamental discriminant Δ . Then A randomly selects $\gamma \in Cl(\Delta)$, an integer $a \leq 2^{160}$, and computes $\alpha = \gamma^a$. A's public key is (Δ, γ, α) , the private key is a.
- **Signature:** To sign the message m, A randomly selects $k \leq 2^{160}$ and computes $\rho = \gamma^k$ and an integer $s = -af(m, \rho) + k$. Then the signature for m is $S = (s, \rho)$.

Verification: B checks that $0 \le s < q$ and that $\gamma^s \alpha^{f(m,\varrho)} = \varrho$.

This surprisingly simple DSA variant appears to be superior to RDSA, for it requires apparently less computation and is based on an allegedly harder problem. However, as in [25], this scheme is secure against existential forgery in a chosen message attack only if af/k is negligible. For example, if a and the output of f are 160 bits wide, and if $1/2^{80}$ is negligible, then k has to be selected 400 bits large (note that s will also have 400 bits). Thus, this DSA variant is considerably less efficient than RDSA.

4.3 The Guillou-Quisquater signature scheme

The Guillou-Quisquater signature scheme [13] works unmodified in our context. For convenience, we present a stripped down version:

- Key generation: A randomly selects a fundamental discriminant Δ . Then A randomly selects $\alpha \in Cl(\Delta)$ and an integer $q \leq 2^{160}$. Then A computes $\theta = \alpha^{-q}$. A's public key is (Δ, q, θ) , the private key is α .
- Signature: To sign the message m, A randomly selects $\kappa \in Cl(\Delta)$ and computes $\varrho = \kappa^q$. Then A computes $\ell = f(m, \varrho)$ and $\sigma = \kappa \alpha^{\ell}$. The signature for m is $S = (\ell, \sigma)$.

Verification: B computes $v = f(m, \sigma^q \theta^\ell)$ and checks that $v = \ell$.

It is obvious that this signature scheme is based on the intractability to compute roots in $Cl(\Delta)$. According to [24], the Guillou-Quisquater signature scheme also can be proven to be secure against existential forgery in a chosen message attack.

5 Efficiency of IQ protocols

In this section, we shall review some algorithms to perform IQ arithmetic, then we shall present some benchmarks for the IQ-RDSA signature scheme and compare these to benchmarks of the RSA signature scheme.

5.1 IQ arithmetic

IQ arithmetic is essentially the same as the composition of binary quadratic forms [8, Sect. 5.2]. The algorithms here are taken from [17]. Recall that integral ideals are represented by pairs (a, b), where a and b are integers, a is positive, and $4a \mid b^2 - \Delta$. For any ideal (a, b) set $c = (b^2 - \Delta)/(4a)$.

First, we present the algorithms for ideal multiplication and ideal squaring. For integers a and b we denote by $(d, x, y) \leftarrow \operatorname{xgcd}(a, b)$ the computation of integers d, x, and y such that d is the greatest common divisor of a and b, and d = ax + by.

The hard part of Algorithms 1 and 2 is the extended Euclidean algorithm (line 1 in both algorithms). Therefore, both algorithms have the best performance, if their input ideals are reduced.

Note that the second extended Euclidean algorithm in Algorithm 1 (line 5) takes on average only a few steps, because d_1 is very small on average.

Next, we present a reduction algorithm, since it is essential to reduce any (intermediate) result. The reduction is performed by Algorithms 3. Algorithm 3 is similar to the extended Euclidean algorithm, and it can be shown that the loop (lines 13 to 22) is executed at most $2 + \lceil \log_2(a/\sqrt{|\Delta|}) \rceil$ times [8, Proposition 5.4.3]. A run time analysis of Algorithm 3 similar to the analysis of the extended Euclidean algorithm shows that Algorithm 3 performs $O(\log^2 |\Delta|)$ steps [5, Sect. 5.6], if the input to this algorithm is the output of either algorithm 1 or 2.

Finally, we note that if (a, b) represents an ideal class, then (a, -b) represents the inverse. If (a, b) is reduced, then so is (a, -b), unless a = b, which happens with negligible probability. For instance, if Δ is a negative odd prime, then there is only one such reduced ideal, and that is (1, 1). Therefore, the inversion of a group element takes almost always constant time. This can be utilized for fast exponentiations of ideals, because one could use a signed-digit exponent recoding [22, Sect. 14.7], which reduces the number of ideal multiplications.

Algorithms 1, 2, and 3 have been implemented using the GNU multiprecision arithmetic library (GMP, version 3.1.1, see http://www.swox.com/gmp/). On a SUN machine with Sparc Ultra II processor (333 MHz), we get the benchmarks in Table 2. From this table, one can conclude that on average an ideal squaring is only slightly faster than an ideal multiplication. Furthermore, the greatest amount of work has to be spent for the reduction, thus any optimization efforts should start here. This could be done by replacing Algorithm 3 by Schönhage's reduction algorithm [29, 33]. Another idea is to use the NUCOMP and NUDUPL algorithms of Shanks [32] for ideal multiplication and squaring. Shanks' Algorithms have the advantage, that most intermediate results are of size $O(\sqrt{|\Delta|})$. Moreover, their outputs are already reduced. This is achieved to the expense of more multiprecision multiplications. But if these algorithms are implemented carelessly, they are still slower than multiplication or squaring with subsequent reduction.

Algorithm 1 Compute $\mathfrak{a}_3 \leftarrow \mathfrak{a}_1 \times \mathfrak{a}_2$ Input: Reduced ideals $\mathfrak{a}_1 = (a_1, b_1)$ and $\mathfrak{a}_2 = (a_2, b_2)$, $\mathfrak{a}, b \subseteq \mathcal{O}_\Delta$ Output: the ideal $\mathfrak{a}_3 = (a_3, b_3)$ 1: $d_1, v, w \leftarrow \operatorname{xgcd}(a_1, a_2)$ 2: $a_3 \leftarrow a_1 a_2$ 3: $b_3 \leftarrow v a_1 (b_2 - b_1)$ 4: if $d_1 \neq 1$ then 5: $d_2, v, w \leftarrow \operatorname{xgcd}(d_1, (b_1 + b_2)/2)$ 6: $a_3 \leftarrow a_3/d_2^2$ 7: $b_3 \leftarrow (b_3 v + w(\Delta - b_1^2)/2)/d_2$ 8: end if 9: $b_3 \leftarrow b_1 + b_3 \mod 2a_3$

Algorithm 2 Compute $\mathfrak{a}_3 \leftarrow \mathfrak{a}_1^2$ Input: A reduced ideal $\mathfrak{a}_1 = (a_1, b_1), \ \mathfrak{a}_1 \subseteq \mathcal{O}_\Delta$ Output: the ideal $\mathfrak{a}_3 = (a_3, b_3)$ 1: $d, v, w \leftarrow \operatorname{xgcd}(a_1, b_1)$ 2: $a_3 \leftarrow (a_1/d)^2$ 3: $b_3 \leftarrow w(\Delta - b_1^2)/(2d)$ 4: $b_3 \leftarrow b_1 + b_3 \mod 2a_3$

Algorithm 3 Reduce a

Input: An ideal $\mathfrak{a} = (a, b), \ \mathfrak{a} \subseteq \mathcal{O}_{\Delta}$ **Output:** A reduced ideal equivalent to \mathfrak{a} 1: if b < 0 then $s \leftarrow -1$ 2: 3: else $s \leftarrow 1$ 4:5: **end if** 6: $b \leftarrow |b|, r \leftarrow b \mod 2a$ 7: if a < r then 8: $b \leftarrow a$ 9: **else** 10: $b \leftarrow r, s \leftarrow -s$ 11: end if 12: $c \leftarrow (b^2 - \Delta)/4a$ 13: while $a > c \, do$ $t_a \leftarrow a, a \leftarrow c$ 14: $q \leftarrow \lfloor b/2a
floor, \ r \leftarrow b mod 2a$ 15: $c \leftarrow t_a - q(r+b)/2$ 16:17:if a < r then $b \leftarrow 2a - r, \ c \leftarrow c + a - r$ 18:else 19: $b \leftarrow r, s \leftarrow -s$ 20: 21:end if 22: end while 23: if s < 0 then $b \leftarrow -b$ 24:25: end if 26: if b > a then $b \leftarrow b - 2a, \, c \leftarrow c - b + a$ 27:28: else if $b \leq -a$ then $b \leftarrow b + 2a, c \leftarrow c + b + a$ 29:30: end if 31: if a = c and b < 0 then $b \leftarrow -b$ 32:33: end if

size of Δ	$\operatorname{multiply}$	square	reduce
512	0.1570	0.1550	0.4455
724	0.2390	0.2295	0.7035
1024	0.3570	0.3380	1.1400
1448	0.5635	0.5230	1.9130
2048	0.9145	0.8340	3.3165
2896	1.4855	1.3475	6.0395
4096	2.4590	2.2270	10.8880

Table 2: Performance of IQ arithmetic (in milliseconds)

Size of	IQ-RDSA				
Δ	Sig	Ver			
687	82.4	245.4			
1208	191.4	554.5			
2084	411.3	1248.8			

Table 3: Some IQ-RDSA timings (in milliseconds)

Table 4: Some RSA timings (in milliseconds)

Size of	RS_{-}	A
n	Sig	Ver
1024	50.7	3.0
2048	342.7	10.7
4096	2447.5	39.8

5.2 Efficiency of some IQ cryptosystems

In this subsection we present benchmarks for the IQ-RDSA signature scheme. This have been implemented using the arithmetic modules that were described in the previous subsection. We have used the following optimizations: For the exponentiation we have used signed-digit exponent recoding. For a signature, we have also used Gordon-Brickell precomputation [22, Sect. 14.6.3] (generated in the key generation step). For the verification we have used simultaneous multiple exponentiation. Some averaged timings are given in Table 3.

For a comparison, we present some benchmarks of the RSA signature scheme in Table 4. We have used the openssl implementation of RSA (openssl version 0.9.6, see http://www.openssl.org/). This implementation makes use of Montgomery exponentiations and Chinese remaindering. As public exponent we have selected the Fermat prime 65537. Note that the parameter size for IQ-RDSA and for RSA has been selected according to Table 1.

The key observation is that an IQ-RDSA signature is eventually faster than a RSA signature. The cross over is at a security level that is rather moderate (according to [20]). Since the efficiency of IQ arithmetic received comparably little attention in the past, it is reasonable to expect significant improvements in the future. These improvements will make IQ cryptosystems even more competitive to traditional cryptosystems.

6 Conclusion

We have given a summary over the selection of the cryptographic parameter for IQ cryptosystems. We have presented IQ versions of some well known protocols, and we have presented efficient algorithms to implement IQ cryptosystems. Finally we have seen that they are efficient and practical. The next steps are: Apply the optimizations to IQ arithmetic as outlined at the end of Sect. 5.1, investigate more protocols (for example, blind signatures, undeniable signatures, signatures with message recovery, and key exchange protocols), and finally, a formal description of IQC similar to [30, 31].

References

- ANSI X9.63. Public Key Cryptography for the Financial Services Industry: Key Agreement and Transport Using Elliptic Curve Cryptography. American Bankers Association, 1999. Working Draft.
- [2] BELLARE, M., AND ROGAWAY, P. Random oracles are practical: a paradigm for designing efficient protocols. In 1st ACM Conference on Computer and Communications Security (1993), pp. 62-73.
- [3] BIEHL, I., BUCHMANN, J., HAMDY, S., AND MEYER, A. A signature scheme based on the intractability of extracting roots. *Designs, Codes and Cryptography* (to appear).
- [4] BOREVICH, Z. I., AND SHAFAREVIC, I. R. Number theory. Academic Press, 1966.
- [5] BUCHMANN, J. Algorithms for binary quadratic forms. http://www.informatik. tu-darmstadt.de/TI/Lehre/WS00_01/Vorlesung/Algorithmische_Zahlentheorie/quf. ps.gz, TU-Darmstadt, FB Informatik, October 2000. Draft.
- BUCHMANN, J., AND WILLIAMS, H. C. A key-exchange system based on imaginary quadratic fields. Journal of Cryptology 1, 3 (1988), 107–118.
- [7] BUELL, D. A. The expectation of success using a Monte Carlo factoring method some statistics on quadratic class numbers. *Mathematics of Computation* 43, 167 (1984), 313–327.
- [8] COHEN, H. A Course in Computational Algebraic Number Theory, vol. 138 of Graduate Texts in Mathematics. Springer-Verlag, 1995.
- [9] COHEN, H., AND LENSTRA, JR., H. W. Heuristics on class groups of number fields. In Number Theory, Noordwijkerhout 1983, H. Jager, Ed., vol. 1068 of Lecture Notes in Mathematics. Springer-Verlag, 1984, pp. 33-62.
- [10] CRAMER, R., AND SHOUP, V. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Advances in Cryptology - CRYPTO '98 (1998), H. Krawczyk, Ed., vol. 1462 of Lecture Notes in Computer Science, Springer-Verlag, pp. 13-25.
- [11] ELGAMAL, T. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31, 4 (1985), 469–472.
- [12] FIPS 186-2. Digital Signature Standard (DSS). NIST, 2000. Federal Information Processing Standards Publication 186-2.
- [13] GUILLOU, L. C., AND QUISQUATER, J.-J. A practical zero-knowledge protocol fitted to security microprocessors minimizing both transmission and memory. In Advances in Cryptology – EUROCRYPT '88 (1988), C. G. Günther, Ed., vol. 330 of Lecture Notes in Computer Science, Springer-Verlag, pp. 123–128.
- [14] HAMDY, S., AND MÖLLER, B. Security of cryptosystems based on class groups of imaginary quadratic orders. In Advances in Cryptology – ASIACRYPT 2000 (2000), T. Okamoto, Ed., vol. 1976 of Lecture Notes in Computer Science, Springer-Verlag, pp. 234–247.
- [15] HORSTER, P., MICHELS, M., AND PETERSEN, H. Meta-ElGamal signature schemes. In 2nd ACM Conference on Computers and Communications Security (1994), ACM Press, pp. 96-107.
- [16] HUA, L. K. Introduction to Number Theory. Springer-Verlag, 1982.

- [17] JACOBSON, JR., M. J. Subexponential Class Group Computation in Quadratic Orders. PhD thesis, Technische Universität Darmstadt, Fachbereich Informatik, Darmstadt, Germany, 1999.
- [18] KOBLITZ, N. Elliptic curve cryptosystems. Mathematics of Computation 48, 177 (1987), 203– 209.
- [19] LANG, S. Algebraic Number Theory, 2 ed., vol. 110 of Graduate Texts in Mathematics. Springer-Verlag, 1994.
- [20] LENSTRA, A. K., AND VERHEUL, E. R. Selecting cryptographic keysizes. In Practice and Theory in Public Key Cryptography, PKC 2000 (2000), H. Imai and Y. Zheng, Eds., vol. 1751 of Lecture Notes in Computer Science, Springer-Verlag, pp. 446-465. Full version available from http://www.cryptosavvy.com/.
- [21] LITTLEWOOD, J. E. On the class number of the corpus $P(\sqrt{-k})$. In Proceedings of the London Mathematical Society, vol. 27 of 2nd series. Cambridge University Press, 1928, pp. 358–372.
- [22] MENEZES, A. J., VAN OORSCHOT, P. C., AND VANSTONE, S. A. Handbook of Applied Cryptography. CRC Press, 1997.
- [23] MILLER, V. S. Use of elliptic curves in cryptography. In Advances in Cryptology CRYPTO '85 (1986), H. C. Williams, Ed., vol. 218 of Lecture Notes in Computer Science, Springer-Verlag.
- [24] POINTCHEVAL, D., AND STERN, J. Security arguments for digital signatures and blind signatures. Journal of Cryptology 13, 3 (2000), 361–396.
- [25] POUPARD, G., AND STERN, J. Security analysis of a practical "on the fly" authentication and siganture generation. In Advances in Cryptology – EUROCRYPT '98 (1998), K. Nyberg, Ed., vol. 1403 of Lecture Notes in Computer Science, Springer-Verlag, pp. 422–436.
- [26] SCHAUB, J. Implementierung von Public-Key-Kryptosystemen über imaginär-quadratischen Ordnungen. Master's thesis, Technische Universität Darmstadt, Fachbereich Informatik, 1999. German.
- [27] SCHNORR, C. P. Efficient signature generation by smart cards. Journal of Cryptology 4, 3 (1991), 161–174.
- [28] SCHNORR, C. P., AND LENSTRA, JR., H. W. A Monte Carlo factoring algorithm with linear storage. *Mathematics of Computation* 43, 167 (1984), 289–311.
- [29] SCHÖNHAGE, A. Fast reduction and composition of binary quadratic forms. In Proc. ISSAC 91 (1991), S. M. Watt, Ed., ACM Press, pp. 128–133.
- [30] SEC 1. *Elliptic Curve Cryptography*. Standards for Efficient Cryptography Group, September 2000. Working Draft. Available from http://www.secg.org/.
- [31] SEC 2. Recommended Elliptic Curve Domain Parameters. Standards for Efficient Cryptography Group, September 2000. Working Draft. Available from http://www.secg.org/.
- [32] SHANKS, D. On Gauss and composition I, II. In Number Theory and Applications, Calgary 1988 (1989), R. A. Mollin, Ed., vol. 265 of NATO ASI Series, Series C, Kluwer Academic Publishers, pp. 163–178, 179–204.
- [33] WEILERT, A. Effiziente Algorithmen zur Berechnung von Idealsummen in quadratischen Ordnungen. PhD thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, 2000. German.