

IQ-CRYPTOGRAPHY PRIMITIVES AND SCHEMES

SAFUAT HAMDY, PAUL DICKINSON

ABSTRACT. We propose public-key primitives and schemes based on IQ-related computational problems for standardisation and deployment. IQ refers to class groups of imaginary quadratic orders, and IQ-cryptography to schemes using these groups. IQ-cryptography offers another secure and efficient alternative to other forms of public key cryptography. Class groups of imaginary quadratic number fields are finite Abelian groups for which several apparently intractable problems exist. These problems include computation of discrete logarithms, roots, and Diffie-Hellman secrets. Therefore, class groups admit to cryptographic schemes of Diffie-Hellman and ElGamal type.

In this article, we present two schemes in the context of IQ-cryptography from a practical standpoint with a view towards including them in a future IEEE standards document. The schemes are a variant of the Schnorr signature scheme as suggested in [16], and the Guillou-Quisquater signature scheme [7]. We also include information on how to implement primitives necessary to perform IQ-cryptography in general. Much of the material in this document is drawn from [9].

1. INTRODUCTION

IQ-cryptography (IQC) refers to cryptographic public-key schemes that are based on class groups of imaginary quadratic orders. Class groups are finite Abelian groups for which several computational problems exist. These problems include the computation of discrete logarithms, element orders, roots, and Diffie-Hellman secrets. In the context of class groups of imaginary number fields, we will denote these problems by IQ-DLP, IQ-OP, IQ-RP and IQ-DHP respectively. Since these problems are apparently intractable in general, class groups are suitable for cryptographic schemes of Diffie-Hellman and ElGamal type. IQ-based cryptographic schemes are precisely those schemes that rely for their security on the apparent intractability of the above-mentioned problems.

1.1. Rationale. The security of commonly used cryptographic schemes (where we mean effectively used schemes such as RSA, DSA, IES, and their elliptic curve variants, for example) is essentially based on three families of computational problems. These are: the integer factoring problem (IFP) and the RSA problem; the Diffie-Hellman problem (FF-DHP) and the discrete logarithm problem (FF-DLP) in multiplicative groups of finite fields (FFs); and the elliptic curve variants thereof (EC-DHP and EC-DLP). For none of these computational problems has a rigorous and unconditional proof of intractability yet been provided. This has repeatedly raised concerns about the security of public-key cryptography. For instance, RSA is so widely used in practice that a major advance in factoring would have a severe impact on the use of secure communications.

Moreover, integer factoring appears to be related to other computational FF problems. In the past, every advance in integer factoring has led to an advance in discrete logarithm computation in finite fields, so it is conceivable that the security of commonly used cryptographic schemes is actually based on only two *independent* families of computational problems (IFP/FF-DLP/FF-DHP etc. and EC-DHP/EC-DLP etc.).

As long as no rigorous unconditional security proofs are available, it is expedient to have as many independent alternatives available as possible. The potential for a breakdown of a cryptographic suite makes it crucial that we have access to well worked out and efficient alternative cryptographic schemes. Alternative schemes also enable one to spread the risk. Besides elliptic curve cryptography, other alternatives such as NTRU are being standardised and deployed. Our objective is to propose the standardization and deployment of IQ-cryptography as another alternative.

A key feature of IQ-cryptography in this context is the observation that its security is apparently independent from other, non-IQ-related computational problems. There are no known connections between any of the IQ-related problems and the computational problems on which the other families of cryptosystems are based. Thus, advances in solving those problems are irrelevant to the security of IQ-cryptography.

The IQ-DLP and the IQ-OP have also the interesting property that the IFP is a complexity theoretic lower bound for these problems, i.e. factoring is not harder than the IQ-DLP or the IQ-OP. Conversely, there is some indication that IQ-related computational problems might be indeed harder than their FF-counterparts or factoring, see Bauer and the first author in [1]. A consequence of this would be that, given a fixed security level, IQ-based schemes are asymptotically faster than traditional factoring- and FF-based schemes.

IQ-related computational problems are hard in general, and suitable cryptographic parameters can be found efficiently. IQ-based cryptographic schemes are efficient if they are used with reasonable parameters, where we mean efficient in practice, not just in theory. Given common levels of security (e.g. 2000 bits for an RSA modulus), IQ-cryptography implementations are currently slower in practice than implementations of traditional or elliptic curve cryptography. However, in many applications public-key cryptography is used to process only very small amounts of data. Therefore, if speed is not a critical issue, IQ-cryptography might be an expedient alternative.

IQ-based cryptographic primitives and schemes are usually variants of standard primitives and schemes (such as Diffie-Hellman key exchange). Consequently, the security proofs of the IQ-based schemes are usually analogous to the security proofs of the traditional schemes. This works for any scheme as long as no special properties of the underlying group is used (such as the knowledge of the group order).

Therefore, given standard notions of security and assuming that the IQ-based computational problems are hard, IQ-based schemes are secure. Moreover, they are independent from other cryptographic primitives, and they are reasonably efficient. Thus, IQ-cryptography provides a valuable contribution to public-key cryptography is, at least, a strong fall back alternative to traditional systems such as RSA.

This document does not include an analysis of these properties, for which we refer the reader to the first author [9]. In this paper, we present two schemes in the context of IQ-cryptography from a practical standpoint, with a view to including them in a future IEEE standards document. The two schemes are a variant of the Schnorr signature scheme [18] (see also [14, Section 11.5.3], see Poupard and Stern [16] for the variant) and the Guillou-Quisquater signature scheme [7].

2. CLASS GROUPS OF IMAGINARY QUADRATIC ORDERS

2.1. Definitions. Here we give a brief overview of class groups of imaginary quadratic orders. An excellent exposition on this subject with further references can be found in Buchmann [4]. A negative integer $\Delta \in \mathbb{Z}$ is called an *imaginary-quadratic discriminant* if it satisfies $\Delta \equiv 0, 1 \pmod{4}$. The *conductor* of an imaginary-quadratic discriminant is the largest integer f for which Δ/f^2 is itself an imaginary-quadratic discriminant.

If $f(\Delta) = 1$, then Δ is called *fundamental*. Note that Δ is fundamental if and only if either $\Delta \equiv 1 \pmod{4}$ and Δ is square-free, or $\Delta \equiv 0 \pmod{4}$ and $\Delta/4 \equiv 2, 3 \pmod{4}$ and $\Delta/4$ is square-free.

The imaginary-quadratic order of discriminant Δ is the \mathbb{Z} -module

$$\mathcal{O}_\Delta = \mathbb{Z} + \frac{\Delta + \sqrt{\Delta}}{2}\mathbb{Z} .$$

\mathcal{O}_Δ is a ring, and every non-null \mathcal{O}_Δ -ideal has a unique representation

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

where $a \in \mathbb{Z}$ and $q \in \mathbb{Q}$ are unique and positive, $b \in \mathbb{Z}$ is unique modulo $2a$, $4a \mid b^2 - \Delta$, and $\gcd(a, b, c) = 1$ for $c = (b^2 - \Delta)/4a$.

If $q \in \mathbb{Z}$, then \mathfrak{a} is an *integral* \mathcal{O}_Δ -ideal, otherwise it is a *fractional* \mathcal{O}_Δ -ideal. Moreover, if $q = 1$, then \mathfrak{a} is a primitive \mathcal{O}_Δ -ideal. The *norm* of a primitive ideal \mathfrak{a} represented by (a, b) is $\text{Norm}(\mathfrak{a}) = a$.

A primitive \mathcal{O}_Δ -ideal \mathfrak{a} is represented by the pair (a, b) . The representation is *normal*, if $-a < b \leq a$. We assume w.l.o.g. that the representation of any primitive ideal is normalised, unless stated otherwise. An integral \mathcal{O}_Δ -ideal is a prime ideal if and only if either q or a is prime. Thus the primitive prime ideals have the form (p, b) with p prime and $p^2 \nmid \Delta$.

Let (a, b) be the normal representation of an \mathcal{O}_Δ -ideal \mathfrak{a} , and let $c = (b^2 - \Delta)/4a$ (note that $c > 0$, since $\Delta < 0$). Then \mathfrak{a} is *reduced* if and only if $a < c$ or $a = c$ and $b \geq 0$. The algorithm for reducing an ideal is given in section 4.1.

We define a relation between \mathcal{O}_Δ -ideals as follow. Two \mathcal{O}_Δ -ideals \mathfrak{a}_1 and \mathfrak{a}_2 are *equivalent* if and only if $\mathfrak{a}_1\mathfrak{a}_2^{-1}$ is a principal \mathcal{O}_Δ -ideal, i.e. $\exists \alpha \in \mathbb{Q}(\sqrt{\Delta})$, $\alpha \neq 0$, such that $\mathfrak{a}_1 = \alpha\mathfrak{a}_2$. We write $\mathfrak{a}_1 \sim \mathfrak{a}_2$ if \mathfrak{a}_1 and \mathfrak{a}_2 are equivalent, otherwise, we write $\mathfrak{a}_1 \not\sim \mathfrak{a}_2$. The relation \sim is an equivalence relation, and the equivalence class of an \mathcal{O}_Δ -ideal \mathfrak{a} is denoted by $[\mathfrak{a}]$ or by \mathfrak{a} .

Multiplication of \mathcal{O}_Δ -ideals is defined as usual and for the product of \mathcal{O}_Δ -ideals \mathfrak{a}_1 and \mathfrak{a}_2 , we write $\mathfrak{a}_1\mathfrak{a}_2$ or $\mathfrak{a}_1 \cdot \mathfrak{a}_2$. An \mathcal{O}_Δ -ideal is *invertible* if there exists an \mathcal{O}_Δ -ideal \mathfrak{a}^{-1} such that $\mathfrak{a}\mathfrak{a}^{-1} \sim \mathcal{O}_\Delta$. Primitive \mathcal{O}_Δ -ideals are always invertible. In particular, for each primitive \mathcal{O}_Δ -ideal $\mathfrak{a} = (a, b)$ there exists an inverse ideal \mathfrak{a}^{-1} that is equivalent to the primitive ideal $(a, -b)$. For the product $\mathfrak{a}_1\mathfrak{a}_2^{-1}$ we also write $\mathfrak{a}_1/\mathfrak{a}_2$. The basis for IQ-cryptography is laid by the following theorem:

Theorem 2.1 (Gauß). *The equivalence classes of \mathcal{O}_Δ form a finite Abelian group under ideal multiplication.*

The group of equivalence classes is called the *class group* and is denoted by $\text{Cl}(\Delta)$; its order is called *class number* and is denoted $h(\Delta)$. The identity element \mathfrak{e} of $\text{Cl}(\Delta)$ is the equivalence class that is represented by the ideal $\mathfrak{e} = (1, \Delta \bmod 2)$.

2.2. Notation. For the remainder of this article, we use the following notations:

- Integers are denoted by Roman letters.
- Δ denotes an imaginary quadratic discriminant.
- Fraktur letters like \mathfrak{a} represent ideals, and class groups are denoted by $\text{Cl}(\Delta)$.
- The equivalence class of an ideal \mathfrak{a} is denoted by its bold equivalent \mathfrak{a} . Thus elements of a class group are denoted by bold fraktur letters.
- We denote the identity element of a class group by \mathfrak{e} .
- If \mathfrak{a} is an element of a class group, then \mathfrak{a} will be understood to be an ideal representing \mathfrak{a} , where we usually assume that \mathfrak{a} is reduced unless stated otherwise.

- If \mathfrak{a} is an ideal, then $\bar{\mathfrak{a}}$ shall be understood to be the equivalent reduced ideal.
- The ideal $\mathfrak{e} = (1, \Delta \bmod 2)$.
- B_x denotes an upper bound on the variable x . These bounds will depend on the security parameter chosen.
- h denotes a cryptographically suitable hash function

$$h : (M, \mathfrak{a}) \rightarrow \{0, \dots, B_h - 1\} .$$

For example, let $\mathfrak{a} = (a, b)$. If $B_h = 2^{160}$, then one could feed the concatenation of M , a , and the sign s of b (e.g. $s = 1$ if $b < 0$, and $s = 0$ otherwise) into SHA1.

3. SECURITY OF IQC

3.1. The Computational Problems. To avoid any ambiguities, we briefly define the computational problems on which the security of IQ-based schemes rest. Let $\text{Cl}(\Delta)$ be a class group. Then we define:

Discrete logarithm problem (IQ-DLP): given $\mathfrak{a}, \mathfrak{b} \in \text{Cl}(\Delta)$, find the smallest positive integer x such that $\mathfrak{b} = \mathfrak{a}^x$ (or decide that no such x exists).

Order problem (IQ-OP): given $\mathfrak{a} \in \text{Cl}(\Delta)$, compute the order $|\langle \mathfrak{a} \rangle|$ of \mathfrak{a} in $\text{Cl}(\Delta)$.

Root problem (IQ-RP): given $\mathfrak{a} \in \text{Cl}(\Delta)$ and an integer $x > 1$, compute \mathfrak{b} such that $\mathfrak{b}^x = \mathfrak{a}$ (or decide that no such $\mathfrak{b} \in \text{Cl}(\Delta)$ exists).

Diffie-Hellman problem (IQ-DHP): given $\mathfrak{g}, \mathfrak{a}, \mathfrak{b} \in \text{Cl}(\Delta)$ with $\mathfrak{a} = \mathfrak{g}^a$ and $\mathfrak{b} = \mathfrak{g}^b$ for some (unknown) integers a and b , compute \mathfrak{g}^{ab} .

If \leq_m denotes deterministic polynomial time reduction, then we clearly have the following computational complexity relations: $\text{IQ-RP} \leq_m \text{IQ-OP} \leq_m \text{IQ-DLP}$ and $\text{IQ-DHP} \leq_m \text{IQ-DLP}$. It is unknown whether these hierarchies are strict or not in the case of class groups, but for practical purposes these problems can be reckoned as equivalent, because there is no method known to compute element orders or Diffie-Hellman secrets without computing first a discrete logarithm; likewise, there is no method known to compute roots in class groups without first computing an element order.

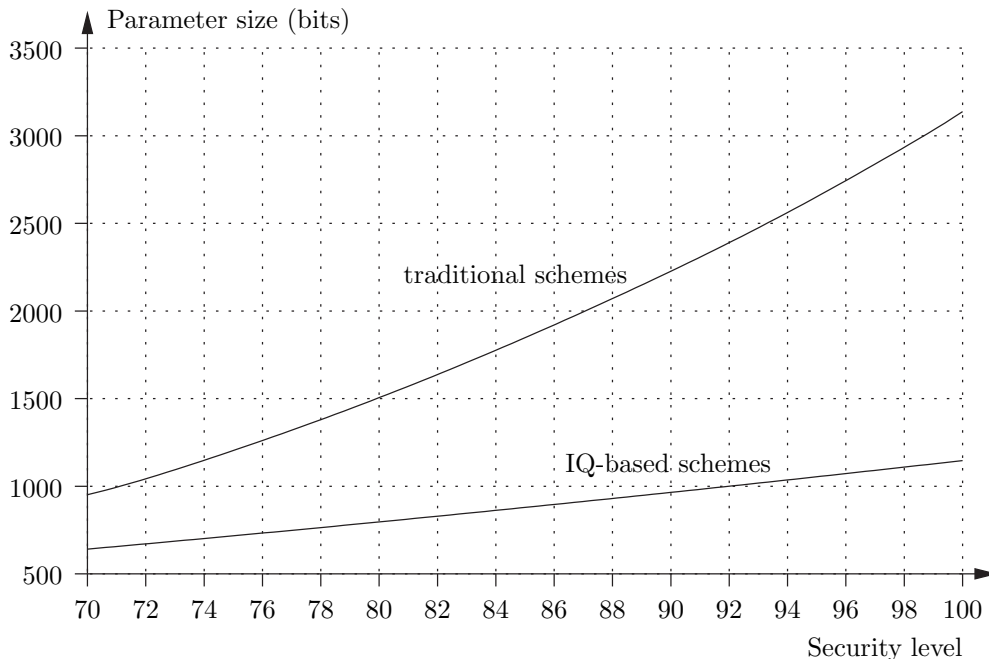
Finally, we have $\text{IFP} \leq_m \text{IQ-OP}$, where IFP denotes the integer factoring problem (this is the foundation of the Schnorr-Lenstra factoring algorithm). Therefore, the complexity to solve the IFP is a lower bound for the complexity to solve the IQ-DLP. There is some indication that this relation is strict, see Bauer and the first author in [1].

The security of the Schnorr scheme variant presented in this paper relies on the IQ-DLP, and the Guillou-Quisquater scheme relies on the IQ-RP. Clearly, the security of the Diffie-Hellman primitive relies on the IQ-DHP. For a more detailed analysis of the difficulty of solving these computational problems see the first author in [9]. We give just a brief sketch of that analysis.

Assuming the truth of the extended Riemann hypothesis and the Cohen-Lenstra heuristics (see e.g. Cohen in [5, Section 5.10.1]) for imaginary quadratic number fields, then it is possible to prove that class groups of imaginary quadratic number fields have good cryptographic properties:

- By the Siegel-Brauer theorem we have (unconditionally) $\lim_{\Delta \rightarrow -\infty} \ln h(\Delta) / \ln \sqrt{|\Delta|} = 1$, so the class number $h(\Delta)$ has asymptotically the order of magnitude of $\sqrt{|\Delta|}$. In fact, the expected size of a randomly chosen fundamental discriminant is $c_3 \sqrt{|\Delta|}$, where $c_3 \approx 0.46$. Moreover, Littlewood proved that $(c_1 +$

FIGURE 3.1. Parameter sizes for IQ-based and traditional cryptographic schemes



$o(1))\sqrt{|\Delta|}/\ln \ln |\Delta| < h(\Delta) < (c_2 + o(1))\sqrt{|\Delta|} \cdot \ln \ln |\Delta|$, where c_1 and c_2 are small constants. Therefore, if $|\Delta|$ is large, then the class group is also large.

- By the Cohen-Lenstra heuristics, we know that with very high probability class groups are cyclic or contain very large cyclic subgroups. Therefore, given a randomly chosen fundamental discriminant, then a randomly chosen element from the according class group generates a very large subgroup with very high probability.
- It follows from the Cohen-Lenstra heuristics that class numbers behave asymptotically like ordinary integers with respect to smoothness probabilities and primality probabilities. Therefore, with very high probability the class number of a randomly chosen fundamental discriminant will contain a very large prime factor.

If the discriminant is sufficiently large, then these properties prevent attacks by means of the Kangaroo algorithm and, with very high probability, of the Pohlig-Hellman algorithm. However, the fastest practical algorithm to compute e.g. a discrete logarithm in a class group is a variant of the MPQS factoring algorithm as described by Jacobson in [11]; we call this algorithm IQ-MPQS. Its expected asymptotic running time is proportional to

$$(3.1) \quad L_{|\Delta|}\left[\frac{1}{2}, c + o(1)\right],$$

where $L_x[\epsilon, c] \stackrel{\text{def}}{=} \exp(c(\ln x)^\epsilon (\ln \ln x)^{1-\epsilon})$ for real ϵ and c with $0 \leq \epsilon \leq 1$ and $c > 0$. Experimental data for the IQ-MPQS strongly suggests that $c = 1$.

Comparing this to the NFS factoring algorithm, which has an expected asymptotic running time proportional to $L_n\left[\frac{1}{3}, c + o(1)\right]$ for $c \approx 1.92$, it becomes obvious that the NFS is asymptotically much faster than the IQ-MPQS. Accordingly, with uniformly growing security level, parameters for integer factoring-based schemes grow much faster

than for IQ-based schemes, see Figure 3.1 (taken from [9]; the estimate for the IQ-based schemes is based on (3.1) with $c = 1$ and data points from [8], the estimate for the traditional schemes is based on the expected running times of the NFS for integer factoring as given by Lenstra and Verheul [13]).

Since the IQ-DLP, IQ-OP, and IQ-RP all appear to be hard problems, these problems provide a solid foundation for public-key cryptography. It is noteworthy that it is even not possible to check efficiently whether the class number is divisible by a particular prime, whether it contains a large prime factor or not, or how large the smooth part of the class number is.

4. ARITHMETIC

In this section we describe some algorithms to perform arithmetic in class groups. We assume that the discriminant is fixed within the context in which these algorithms operate. Thus, the discriminant will not be explicitly listed as input parameter; instead, it shall be understood to be an implicit input parameter to all algorithms below.

4.1. IQ-Reduce (IQReduce). The elements of a class group are equivalence classes, i.e. sets of ideals. In order to compute in class groups, one must choose a representative. In order to decide which representative for each equivalence class we choose, we use the following theorem:

Theorem 4.1. *Let \mathcal{O}_Δ be an imaginary quadratic order. Then for every \mathcal{O}_Δ -ideal \mathfrak{a} there exists a unique reduced primitive ideal $\bar{\mathfrak{a}}$ that is equivalent to \mathfrak{a} .*

Thus, we represent each equivalence class by the unique reduced primitive ideal of that class. This representation is space efficient, because if \mathfrak{a} is reduced and (a, b) is its normalised representation, then $a < \sqrt{|\Delta|/3}$, and since $|b| \leq a$, the binary representation of \mathfrak{a} is at most as large as the binary representation of Δ .

However, the product of reduced ideals is usually not reduced. Therefore, given an ideal \mathfrak{a} , we need to compute the reduced equivalent ideal $\bar{\mathfrak{a}}$ (in normalised representation). The primary reason for doing this is for efficiency of multiplication and storage. If ideals are not reduced, then their representations will become very large. Before we present the reduction algorithm, we first give an algorithm to normalise the representation of an ideal.

Algorithm 1 IQNorm

Input: The representation (a, b) of an ideal \mathfrak{a} .

Output: The normalised representation of \mathfrak{a} .

```

1: if  $-a < b \leq a$  then return  $(a, b)$ 
2:  $s \leftarrow \lfloor (a - b)/2a \rfloor$ 
3:  $b \leftarrow b + 2sa$ 
4: return  $(a, b)$ 

```

The reduction algorithm below will make heavy use of the normalisation algorithm. However, the reduction algorithm operates not just on (a, b) but on triples (a, b, c) , where $c = (b^2 - \Delta)/4a$. We may call this triple the extended representation of an ideal. (The triple actually represents the according positive definite binary quadratic form.) The extended representation of an ideal will always be used for computations only, such as IQReduce below, but never for external representation (i.e. for storage or transmission). The following algorithm normalises an ideal in extended representation efficiently:

Algorithm 2 IQXNorm

Input: The extended representation (a, b, c) of an ideal \mathfrak{a} .**Output:** The normalised extended representation of \mathfrak{a} .

```

1:  $q \leftarrow \lfloor (a - b)/2a \rfloor$ ,  $r \leftarrow (a - b) \bmod 2a$ 
2: if  $q \neq 0$  then
3:    $t_b \leftarrow a - r$ 
4:    $t_c \leftarrow (b + t_b)/2$ 
5:    $b \leftarrow t_b$ ,  $c \leftarrow c - qt_c$ 
6: end if
7: return  $(a, b, c)$ 

```

Now we can give a reduction algorithm as it is typically implemented in many programming libraries.

Algorithm 3 IQReduce

Input: An \mathcal{O}_Δ -ideal (a, b) .**Output:** The equivalent reduced ideal.

```

1:  $(a, b) \leftarrow \text{IQNorm}(a, b)$ 
2:  $c \leftarrow (b^2 - \Delta)/4a$ 
3: while  $a > c$  do
4:    $(a, b, c) \leftarrow \text{IQXNorm}(c, -b, a)$ 
5: end while
6: if  $b < 0$  and  $a = c$  then  $b \leftarrow -b$ 
7: return  $(a, b)$ 

```

This algorithm is related to the ordinary extended Euclidean algorithm. In fact, its running time analysis is very similar to that of the extended Euclidean algorithm; the running time of IQReduce is proportional to $\log^2 |\Delta|$. There are other variants of IQReduce as given above. One variant is due to Rickert [17], which has also quadratic running time but in practice it is faster than IQReduce (the ideas of Rickert for reduction resemble that of Lehmer for the extended Euclidean algorithm). Another variant is due to Schönhage [19], which is asymptotically much faster but unsuitable for practical purposes.

4.2. IQ-Multiplication (IQMultiply). Given two ideals \mathfrak{a} and \mathfrak{b} , we wish to compute the (reduced ideal that is equivalent to the) product $\mathfrak{a}\mathfrak{b}$. This is a fundamental operation for any IQ-algorithm. The following algorithm is taken from Jacobson [11], and exploits the special case where $\gcd(a_1, a_2) = 1$. Multiplication using this method depends on the implementation of the gcd function. We assume that the implementer has access to an xgcd function as follows

Algorithm 4 xgcd

Input: Integers a and b .**Output:** Integers d , x , and y such that $d = \gcd(a, b)$ and $ax + by = d$.

Since ideal products are not reduced in general, we have to reduce the result of a multiplication. The following algorithm takes care of that at the end.

Algorithm 5 IQMultiply

Input: \mathcal{O}_Δ -ideals $\mathfrak{a}_1 = (a_1, b_1)$ and $\mathfrak{a}_2 = (a_2, b_2)$.**Output:** The reduced \mathcal{O}_Δ -Ideal $\mathfrak{a}_3 = (a_3, b_3)$ equivalent to $\mathfrak{a}_1 \cdot \mathfrak{a}_2$.

```

1:  $d_1, v, w \leftarrow \text{xgcd}(a_1, a_2)$ 
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:    $d_2, v, w \leftarrow \text{xgcd}(d_1, (b_1 + b_2)/2)$ 
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:  $b_3 \leftarrow b_1 + b_3$ 
10: return IQReduce( $a_3, b_3$ )

```

The algorithm to compute the reduced ideal that is equivalent to the quotient of two (primitive) ideals is similar. Specifically, $\mathfrak{a}_1/\mathfrak{a}_2$ means $\mathfrak{a}_1 \mathfrak{a}_2^{-1}$, and if a primitive ideal \mathfrak{a} is represented by (a, b) , then the ideal \mathfrak{a}^{-1} is equivalent to the ideal represented by $(a, -b)$. Moreover, if the ideal represented by (a, b) is reduced, then so is the ideal represented by $(a, -b)$, and if (a, b) is normal, then so is $(a, -b)$ unless $a = b$, in which case $\mathfrak{a} \sim \mathfrak{a}^{-1}$. About this one does not need to take care for intermediate results. Therefore, computing (the reduced ideal equivalent to) the inverse of a reduced ideal means to flip the sign of b .

Steps 1 to 9 of IQMultiply can be replaced by Shanks' NUCOMP, an algorithm which yields an almost reduced ideal that is equivalent to the product of its input ideals. If NUCOMP is carefully implemented, then IQMultiply with NUCOMP is considerably faster than IQMultiply as above. See van der Poorten and Jacobson [12, 20] for a comprehensive description of NUCOMP.

4.3. IQ-Square (IQSquare). This primitive can be used instead of IQMultiply for the case when an ideal is being squared. This algorithm is more efficient in that case. It too is taken from Jacobson [11].

Algorithm 6 IQSquare

Input: An \mathcal{O}_Δ -ideal $\mathfrak{a}_1 = (a_1, b_1)$.**Output:** The reduced \mathcal{O}_Δ -ideal $\mathfrak{a}_3 = (a_3, b_3)$ equivalent to \mathfrak{a}_1^2 .

```

1:  $d, v, w \leftarrow \text{xgcd}(a_1, b_1)$ 
2:  $a_3 \leftarrow (a_1/d)^2$ 
3:  $b_3 \leftarrow w(\Delta - b_1^2)/(2d)$ 
4:  $b_3 \leftarrow b_1 + b_3$ 
5: return IQReduce( $a_3, b_3$ )

```

Steps 1 to 4 of IQSquare can be replaced by Shanks' NUDUPL, an algorithm which yields an almost reduced ideal that is equivalent to the square of its input ideal. If NUDUPL is carefully implemented, then IQSquare with NUDUPL is considerably faster than IQSquare as above. See van der Poorten and Jacobson [12, 20] for a comprehensive description of NUDUPL.

4.4. IQ-Exponentiate (IQExp). Exponentiation is a frequently used operation in IQ-cryptography. It is done using the same square and multiply algorithm as with the integers, using the operations IQMultiply and IQSquare as above. As explained above, inverting a reduced ideal is a trivial operation. This can be exploited for exponentiation methods that involve signed exponent recoding.

Generic algorithms can be found in [14, Sections 14.6 and 14.7]. We state here just a straight forward interface. Some implementations use tables of precomputed ideal powers; in such cases these tables are understood to be additional input parameters.

Algorithm 7 IQExp

Input: An \mathcal{O}_Δ -ideal \mathfrak{a} and an integer e .

Output: The reduced \mathcal{O}_Δ -ideal \mathfrak{b} equivalent to \mathfrak{a}^e .

4.5. IQ-Random (IQRand). Here we give a procedures for choosing a random element in a class group of imaginary quadratic order. For an extended treatment, see [8]. By a random element, we mean a random reduced primitive \mathcal{O}_Δ -ideal $\mathfrak{a} = (a, b)$.

Algorithm 8 IQRand

Input: Parameters B_k , B_p , and B_e .

Output: A random reduced primitive ideal \mathfrak{a} .

```

 $\mathfrak{a} \leftarrow (1, \Delta \bmod 2)$ 
for  $k \leftarrow 1$  to  $B_k$  do
  repeat
     $p \stackrel{\text{rnd}}{\leftarrow} \{1, \dots, B_p\}$ 
  until  $p$  is prime and  $\left(\frac{\Delta}{p}\right) = 1$ 
   $b \leftarrow \Delta^{1/2} \bmod p$  (e.g. using Shanks' algorithm)
   $\mathfrak{p} \leftarrow (p, b)$ 
   $e \stackrel{\text{rnd}}{\leftarrow} \{1, \dots, B_e\}$ 
   $\mathfrak{a} \leftarrow \text{IQMultiply}(\mathfrak{a}, \text{IQExp}(\mathfrak{p}, e))$ 
end for
return  $\mathfrak{a}$ 

```

The parameters B_k , B_p and B_e can be chosen such that the output of IQRand is uniformly distributed. However, if random elements are to be chosen very efficiently (e.g. for the generation of a Guillou-Quisquater signature), then this choice is unreasonable. The output distribution of IQRand is not known to be distinguishable from the uniform distribution in practice with the following parameter choices: Fix $B_e = 1$ and choose B_k and B_p such that $B_p^{B_k} > |\Delta|$. If B_p is fixed, then $B_k = \lceil \log |\Delta| / \log B_p \rceil$. For instance, on a 64 bit CPU $B_p = 2^{64} - 1$ leads to efficient implementations where the prime finding process can be done with single precision arithmetic provided by the CPU.

5. PRIMITIVES FOR THE SELECTION OF A CLASS GROUP

The first step in any IQ-scheme is to choose a particular class group to use. Any discriminant determines a unique class group. Therefore, we represent a class group by the discriminant Δ of the respective order. We generate Δ based on work in [9]

which describes how to choose a cryptographically suitable discriminants. We call a discriminant cryptographically suitable if the expected time to solve the computational problems on which we base IQC in the respective class group exceeds a predetermined limit with very high probability. Specifically, let t be a security parameter and let $\ell_\Delta(t)$ a function that outputs an integer ℓ_Δ such that computing e.g. a discrete logarithm in the respective class group of a fundamental imaginary quadratic discriminant of length ℓ_Δ is expected to require time proportional to 2^t .

5.1. IQ-Generate Group (IQGG). We give three algorithms for generating an appropriate group. We subsume them under the name IQGG. We assume that the implementer can check integers for primality.

5.1.1. IQGG-1odd and IQGG-1even. The first two methods for generating a group require the generation of only one prime p . These methods are slower than IQGG-2 (see below), since they require that one generates a much larger prime than IQGG-2. However, choosing a new class group will usually be done very infrequently, and in most cases speed will not be an issue. Technically it is sufficient to choose one IQGG method once and forever (e.g. IQGG-1odd). We present several methods because they allows access to a larger possible number of class groups.

Algorithm 9 IQGG-1odd

Input: An integer $\ell_\Delta > 3$.

Output: A fundamental imaginary quadratic discriminant Δ of length ℓ_Δ .

```

1: repeat
2:    $p \xleftarrow{\text{rnd}} \{2^{\ell_\Delta-3}, \dots, 2^{\ell_\Delta-2} - 1\}$ 
3:    $p \leftarrow 4p + 3$ 
4: until  $p$  is prime
5:  $\Delta \leftarrow -p$ 
6: return  $\Delta$ 

```

Algorithm 10 IQGG-1even

Input: An integer $\ell_\Delta > 5$.

Output: A fundamental imaginary quadratic discriminant Δ of length ℓ_Δ .

```

1: repeat
2:    $p \xleftarrow{\text{rnd}} \{2^{\ell_\Delta-5}, \dots, 2^{\ell_\Delta-4} - 1\}$ 
3:    $p \leftarrow 4p + 1$ 
4: until  $p$  is prime
5:  $\Delta \leftarrow -4p$ 
6: return  $\Delta$ 

```

5.1.2. IQGG-2. Another method for generating a group requires two primes, p and q . In practice, this algorithm is faster than IQGG-1odd and IQGG-1even as the length of primes p and q can be approximately half that used in IQGG-2. Further, it can be shown that IQ-DLP is at least as hard as Integer Factoring in class groups with discriminants of this type. (There is no reason to believe that IQ-DLP is any easier in class groups with discriminants found by IQGG-1odd or IQGG-1even though.)

Algorithm 11 IQGG-2**Input:** An integer ℓ_Δ **Output:** A fundamental imaginary quadratic discriminant Δ of length ℓ_Δ .

```

1:  $\ell_p \leftarrow \lceil \ell_\Delta / 2 \rceil$ ,
2: repeat
3:    $p \xleftarrow{\text{rnd}} \{2^{\ell_p-2}, \dots, 2^{\ell_p-1} - 1\}$ 
4:    $p \leftarrow 2p + 1$ 
5: until  $p$  is prime
6: repeat
7:    $q \xleftarrow{\text{rnd}} \{ \lceil (\lceil 2^{\ell_\Delta-1} / p \rceil - 1) / 2 \rceil, \dots, \lfloor (\lfloor 2^{\ell_\Delta} / p \rfloor - 1) / 2 \rfloor \}$ 
8:    $q \leftarrow 2q + 1$ 
9: until  $p \not\equiv q \pmod{4}$  and  $q$  is prime
10:  $\Delta \leftarrow -pq$ 
11: return  $\Delta$ 

```

Note that in the case of a composite discriminant Δ it must be stored and transmitted as a list of its prime factors (here the pair of p and q) in order to facilitate checks on Δ .

6. PRIMITIVES BASED ON THE IQ-DLP AND THE IQ-DHP

In this section we describe some primitives based on the IQ-DLP and the IQ-DHP.

6.1. IQ-DLP and IQ-DHP Domain Parameters. The domain parameters for IQ-DLP and IQ-DHP primitives are structurally identical. We refer to both of them by $\mathcal{D}^{\text{IQDL}}$. This is the triple $\mathcal{D}^{\text{IQDL}} = (t, \Delta, \mathbf{b})$, where t denotes a security parameter, Δ denotes an imaginary quadratic discriminant, and $\mathbf{b} = (a, b)$ denotes the normal representation of a reduced \mathcal{O}_Δ -ideal.

A domain parameter $\mathcal{D}^{\text{IQDL}}$ is valid if and only if:

- t is acceptable (according to some external policy).
- The integer Δ is indeed a fundamental imaginary quadratic discriminant that has the size according to t . Note that if Δ is composite, then it must be represented by its prime factorisation in order to facilitate an efficient check of fundamentality. For simplicity we don't discuss the technical ramifications thereof in this article.
- The pair (a, b) is indeed the normal representation of a valid reduced primitive \mathcal{O}_Δ -ideal that is neither the null ideal nor a principal ideal, see Section 2.1.

Valid IQ-DLP domain parameters may be generated with the following primitive:

Algorithm 12 IQDL-DPGP**Input:** A security parameter t .**Output:** The domain parameters $\mathcal{D}^{\text{IQDL}} = (t, \Delta, \mathbf{b})$.

```

 $\Delta \leftarrow \text{IQGG}(\ell_\Delta(t))$ 
repeat
   $\mathbf{b} \leftarrow \text{IQRand}()$ 
until  $\text{Norm}(\mathbf{b}) \neq 1$ 
return  $(t, \Delta, \mathbf{b})$ 

```

6.2. IQ-DLP and IQ-DHP Key Pairs. The key pairs for IQ-DLP and IQ-DHP primitives are structurally identical. We refer to both of them by $\mathcal{K}^{\text{IQDL}} = (\mathcal{K}_{\text{pub}}^{\text{IQDL}}, \mathcal{K}_{\text{priv}}^{\text{IQDL}})$. A key pair is only meaningful in the context of specific domain parameters. Given valid IQ-DLP domain parameters $\mathcal{D}^{\text{IQDL}} = (t, \Delta, \mathfrak{b})$, then we have for the private key $\mathcal{K}_{\text{priv}}^{\text{IQDL}} = a$, where $1 < a < 2^{2t}$ is a randomly chosen integer, and for the public key $\mathcal{K}_{\text{pub}}^{\text{IQDL}} = \mathfrak{a}$, where $\mathfrak{a} \sim \mathfrak{b}^a$.

A public key $\mathcal{K}_{\text{pub}}^{\text{IQDL}} = \mathfrak{a} = (a, b)$ is valid with respect to a particular domain parameter $(t, \Delta, \mathfrak{b})$ if and only if the pair (a, b) is indeed the normal representation of a valid reduced primitive \mathcal{O}_{Δ} -ideal that is neither the null ideal nor a principal ideal, see Section 2.1.

A valid IQ-DLP key pair may be generated with the following primitive:

Algorithm 13 IQDL-KGP

Input: Domain parameters $\mathcal{D}^{\text{IQDL}} = (t, \Delta, \mathfrak{b})$.

Output: A key pair $\mathcal{K}^{\text{IQDL}} = (\mathcal{K}_{\text{priv}}^{\text{IQDL}}, \mathcal{K}_{\text{pub}}^{\text{IQDL}})$.

```

repeat
   $a \xleftarrow{\text{rnd}} \{2, 3, \dots, 2^{2t}\}$ 
   $\mathfrak{a} \leftarrow \text{IQExp}(\mathfrak{b}, a)$ 
until  $\text{Norm}(\mathfrak{a}) \neq 1$ 
return  $(a, \mathfrak{a})$ 

```

6.3. Primitives for the IQ-DH key exchange scheme. Diffie Hellman secrets can be exchanged using IQ-cryptography in much the same manner as in the integers. Below we present the basic two-party Diffie-Hellman primitive.

6.3.1. The IQ-Diffie-Hellman Secret Value Derivation Primitive (IQDL-SVDP-DH).

We assume that suitable domain parameters $\mathcal{D}^{\text{IQDL}}$ have already been agreed on, key pairs have been established, and each party is respectively in possession of the other participants purported public key.

Algorithm 14 IQDL-SVDP-DH

Input: Common valid domain parameters $\mathcal{D}^{\text{IQDL}} = (t, \Delta, \mathfrak{b})$;

the own key pair $\mathcal{K}^{\text{IQDL}} = (a, \mathfrak{a})$;

the other participant's purported public key \mathfrak{a}' .

Output: A common shared secret ideal \mathfrak{c} .

```

1:  $\mathfrak{c} \leftarrow \text{IQExp}(\mathfrak{a}', a)$ 
2: return  $\mathfrak{c}$ 

```

6.4. Primitives for the IQ-Schnorr scheme. Here we present the signature primitive of a Schnorr variant for use with IQ-cryptography. It can be invoked in a signature scheme with appendix. Since the class number is unknown, we cannot reduce exponents modulo the class number or divisors thereof. This variant is based on the equivalent scheme for multiplicative groups of finite rings by Poupard and Stern [16]. The security of IQ-Schnorr is dependent on the IQ-DLP. Let h be a suitable cryptographic hash function that maps two bit strings of arbitrary length to bit strings of length $2t$.

6.4.1. *The IQ-Schnorr Signature Primitive (IQDL-SP-Schnorr)*. The following primitive generates an IQ-Schnorr signature:

Algorithm 15 IQDL-SP-Schnorr

Input: Valid domain parameters $\mathcal{D}^{\text{IQDL}} = (t, \Delta, \mathbf{b})$;
the own key pair $\mathcal{K}^{\text{IQDL}} = (a, \mathbf{a})$;
a message M .
Output: A signature $S^{\text{IQ-Schnorr}}(M) = (s, r)$ on M .

- 1: $(k, \mathfrak{k}) \leftarrow \text{IQDL-KGP}((\frac{5}{2}t, \Delta, \mathbf{b}))$
 - 2: $r \leftarrow h(M, \mathfrak{k})$
 - 3: $s \leftarrow -ar + k$ //Note: No modular reduction of s
 - 4: **return** (r, s)
-

6.4.2. *IQ-Schnorr Verification Primitive (IQDL-VP-Schnorr)*. The following primitive checks the validity of an IQ-Schnorr signature:

Algorithm 16 IQDL-VP-Schnorr

Input: Valid domain parameters $\mathcal{D}^{\text{IQDL}} = (t, \Delta, \mathbf{b})$;
the signer's purported public key $\mathcal{K}_{\text{pub}}^{\text{IQDL}} = \mathbf{a}$;
a message M ;
a signature $S^{\text{IQ-Schnorr}} = (r, s)$.
Output: valid or invalid.

- 1: $\mathbf{v} \leftarrow \text{IQMultiply}(\text{IQExp}(\mathbf{b}, s), \text{IQExp}(\mathbf{a}, r))$
 - 2: **if** $h(M, \mathbf{v}) = r$ **then return** valid
 - 3: **else return** invalid
-

6.4.3. *Security of the IQ-Schnorr Primitives*. If $B_h B_a B_n / B_k$ and $1/B_h$ and $1/B_a$ are all negligible, then the above Schnorr scheme is secure against existential forgery using an adaptively chosen message attack [16]. The security proof is given in the Random Oracle Model [2]. It follows that B_k has to be much larger compared to the traditional Schnorr or DSA schemes. In particular, if 2^{-t} is deemed to be negligible, then one usually finds $B_h = B_a = 2^{2t}$, and thus $B_k \geq 2^{5t}$ must hold. For instance, if 2^{-80} is deemed negligible, then $B_h = B_a = 2^{160}$, and thus $B_k \geq 2^{400}$.

6.4.4. *Efficiency of the IQ-Schnorr Primitives*. This has an obvious impact on the efficiency of the scheme, since k has to be (at least) $5t$ bits wide, while the traditional Schnorr or DSA scheme would require k to be only $2t$ bits wide. Since s has the same order of magnitude as k , it follows that the signer as well as the verifier is burdened with more computational work compared to traditional DSA. One can partially compensate for this in the signing step by a suitable multi-exponentiation method with a few precomputations [9].

6.5. **Encryption and Decryption Primitives: A Digression.** We will not propose encryption and decryption primitives here. Like the IQ-DH primitive, it would be easy to remodel the IES encryption scheme and its primitives, or even the Shoup-Cramer encryption scheme and its primitives for class groups. However, it is debatable whether this is really necessary. Public-key encryption is almost always used to secure very small amounts of data, namely random session keys for symmetric encryption. Given that one

never cares about which particular random session key is used, only that it is perfectly random (in a suitable sense), the idea of a serialised Diffie-Hellman key exchange seems attractive:

Algorithm 17 IQDL-EP

Input: Valid domain parameters $\mathcal{D}^{\text{IQDL}} = (t, \Delta, \mathbf{b})$;
the recipient's purported public key $\mathcal{K}_{\text{pub}}^{\text{IQDL}} = \mathbf{a}$.

Output: A secret random ideal \mathbf{c} ;
the encryption $E^{\text{IQDL}} = \mathfrak{d}$ of \mathbf{c} .

- 1: $(d, \mathfrak{d}) \leftarrow \text{IQDL-KGP}(\mathcal{D}^{\text{IQDL}})$
 - 2: $\mathbf{c} \leftarrow \text{IQDL-SVDP-DH}(\mathcal{D}^{\text{IQDL}}, (d, \mathfrak{d}), \mathbf{a})$
 - 3: **return** \mathbf{c}, \mathfrak{d}
-

Note that no particular message (i.e. session key) is being encrypted. Instead, the encrypter initiates a Diffie-Hellman key exchange (using the recipients domain parameters and public key), generates her own ephemeral key pair and completes her part of the exchange. The ideal \mathbf{c} is converted to a session key using a suitable key derivation function; this session key is used to encrypt a message using a symmetric cipher. The ephemeral public key \mathfrak{d} is attached to the encryption; it can be viewed as encryption of \mathbf{c} . It is debatable whether this can still be called public-key encryption.

This design has the great advantage of a simple security proof. Compare this design to that of Ferguson and Schneier in [6, Section 13.6].

The corresponding “decryption” primitive works as follows:

Algorithm 18 IQDL-DP

Input: Valid domain parameters $\mathcal{D}^{\text{IQDL}} = (t, \Delta, \mathbf{b})$;
the own key pair $\mathcal{K}^{\text{IQDL}} = (a, \mathbf{a})$;
the encryption $E^{\text{IQDL}} = \mathfrak{d}$ of a secret random ideal.

Output: A shared secret ideal.

- 1: $\mathbf{c} \leftarrow \text{IQDL-SVDP-DH}(\mathcal{D}^{\text{IQDL}}, \mathcal{K}^{\text{IQDL}}, \mathfrak{d})$
 - 2: **return** \mathbf{c}
-

Again, the ideal \mathbf{c} is converted to a session key using a suitable key derivation function; that key is used for a symmetric cipher.

7. PRIMITIVES BASED ON THE IQ-RP

7.1. IQ-RP Domain Parameters. In this section we describe some primitives based on the IQ-RP.

We refer to the domain parameters of for the IQ-RP by \mathcal{D}^{IQR} . This is the triple $\mathcal{D}^{\text{IQR}} = (t, \Delta, a)$, where t denotes a security parameter, Δ denotes an imaginary quadratic discriminant, and n denotes an integer.

A domain parameter \mathcal{D}^{IQR} is valid if and only if:

- t is acceptable (according to some external policy).

- The integer Δ is indeed a fundamental imaginary quadratic discriminant that has the size according to t . Note that if Δ is composite, then it must be represented by its prime factorisation in order to facilitate an efficient check of fundamentality. For simplicity we don't discuss the technical ramifications thereof in this article.
- The integer a is in the range $1 < a < 2^{2t}$.

Valid IQ-RP domain parameters may be generated with the following primitive:

Algorithm 19 IQR-DPGP

Input: A security parameter t .

Output: The domain parameters $\mathcal{D}^{\text{IQDL}} = (t, \Delta, a)$.

```

 $\Delta \leftarrow \text{IQGG}(\ell_{\Delta}(t))$ 
 $a \xleftarrow{\text{rnd}} \{2, 3, \dots, 2^{2t}\}$ 
return  $(t, \Delta, a)$ 

```

7.2. IQ-RP Key Pairs. We refer to the key pairs for the IQ-RP by $\mathcal{K}^{\text{IQR}} = (\mathcal{K}_{\text{pub}}^{\text{IQR}}, \mathcal{K}_{\text{priv}}^{\text{IQR}})$. A key pair is only meaningful within the context of specific domain parameters. Given valid IQ-RP domain parameters $\mathcal{D}^{\text{IQR}} = (t, \Delta, a)$, then we have for the private key $\mathcal{K}_{\text{priv}}^{\text{IQR}} = \mathfrak{b}$, where \mathfrak{b} is a randomly chosen primitive and reduced \mathcal{O}_{Δ} -ideal in normal representation, and for the public key $\mathcal{K}_{\text{pub}}^{\text{IQR}} = \mathfrak{a}$, where $\mathfrak{a} \sim \mathfrak{b}^a$.

A public key $\mathcal{K}_{\text{pub}}^{\text{IQR}} = \mathfrak{a} = (a, b)$ is valid with respect to a particular domain parameter (t, Δ, a) if and only if the pair (a, b) is indeed the normal representation of a valid primitive and reduced \mathcal{O}_{Δ} -ideal that is neither the null ideal nor a principal ideal, see Section 2.1.

A valid IQ-RP key pair may be generated with the following primitive:

Algorithm 20 IQR-KGP

Input: Domain parameters $\mathcal{D}^{\text{IQDL}} = (t, \Delta, a)$.

Output: A key pair $\mathcal{K}^{\text{IQR}} = (\mathcal{K}_{\text{priv}}^{\text{IQR}}, \mathcal{K}_{\text{pub}}^{\text{IQR}})$.

```

repeat
   $\mathfrak{b} \leftarrow \text{IQRand}()$ 
   $\mathfrak{a} \leftarrow \text{IQExp}(\mathfrak{b}, a)$ 
until  $\mathfrak{a}, \mathfrak{b} \neq \mathfrak{e}$ 
return  $(\mathfrak{b}, \mathfrak{a})$ 

```

7.3. Primitives for the IQ-Guillou-Quisquater signature scheme. Here we present the Guillou-Quisquater signature primitive for use with IQ-cryptography. It can be invoked in a signature scheme with appendix. The security of IQ-GQ is dependent on the IQ-RP. Let h be a suitable cryptographic hash function that maps two bit strings of arbitrary length to bit strings of length $2t$.

7.3.1. The IQ-Guillou-Quisquater Signature Primitive (IQR-SP-GQ). The following primitive generates an IQ-Schnorr signature:

Algorithm 21 IQR-SP-GQ

Input: Valid domain parameters $\mathcal{D}^{\text{IQR}} = (t, \Delta, a)$;
the own key pair $\mathcal{K}^{\text{IQR}} = (\mathbf{b}, \mathbf{a})$;
a message M .

Output: A signature $S^{\text{IQ-GQ}}(M) = (s, \mathfrak{s})$ on M .

- 1: $(\mathbf{c}, \mathfrak{d}) \leftarrow \text{IQR-KGP}(\mathcal{D}^{\text{IQR}})$
 - 2: $s \leftarrow h(M, \mathfrak{d})$
 - 3: $\mathfrak{s} \leftarrow \text{IQMultiply}(\mathbf{c}, \text{IQExp}(\mathbf{a}, -s))$
 - 4: **return** (s, \mathfrak{s})
-

7.3.2. *The IQ-Guillou-Quisquater Verification Primitive (IQR-VP-GQ).* The following primitive checks the validity of an IQ-GQ signature:

Algorithm 22 IQR-VP-GQ

Input: Valid domain parameters $\mathcal{D}^{\text{IQR}} = (t, \Delta, a)$;
the signer's purported public key $\mathcal{K}_{\text{pub}}^{\text{IQR}} = \mathbf{a}$;
a message M ;
a signature $S^{\text{IQ-GQ}} = (s, \mathfrak{s})$.

Output: valid or invalid.

- 1: $\mathbf{v} \leftarrow \text{IQMultiply}(\text{IQExp}(\mathfrak{s}, a), \text{IQExp}(\mathbf{a}, s))$
 - 2: **if** $h(M, \mathbf{v}) = s$ **then return** valid
 - 3: **else return** invalid
-

7.3.3. *Security of the IQ-GQ Primitives.* The Guillou-Quisquater signature scheme is derived from a zero-knowledge identification scheme, and the security proof for the signature scheme carries over to the class group case in a straight forward way. It is of the same type as in Pointcheval and Stern [15], thus, the security proof for the Guillou-Quisquater signature scheme is not tight as in Bellare and Rogaway [3]. The security proof for the Guillou-Quisquater signature scheme is also to be understood in the Random Oracle Model [2].

7.3.4. *Efficiency of the IQ-GQ Primitives.* The signer has to perform only two exponentiations. However, one has to choose a random element, which is not a trivial task; without precomputations, choosing a random element in class groups is as expensive as an exponentiation.

8. IQC SCHEMES

We can proceed here as in IEEE P1363 [10, Sections 9 and 10]. Specifically, we can devise the schemes IQDLKAS-DH1 and IQDLKAS-DH2 in analogy to DL/ECKAS-DH1 and DL/ECKAS-DH2, and the schemes IQDLSSA-Schnorr and IQRSSA-GQ in analogy to DL/ECSSA and IFSSA.

8.1. **IQDLKAS-DH1.** We show here how to devise IQDLKAS-DH1 (we omit some technical details, though). Compare this to IEEE P1363 [10, Section 9.2].

8.1.1. *Scheme Options.* IEEE P1363 specifies several Diffie-Hellman key agreement primitives for finite fields and elliptic curves. Apart from the ordinary version, there is the version with cofactor exponentiation, and MQV version. These require the knowledge of the group order, therefore, these are not available for class groups. Therefore, IQDL-SKDP-DH as described above is the only choice. If we model the scheme options after [10, Section 9.2.1], then the only remaining option is the choice of a key derivation function, which could be KDF1 from IEEE P1363 [10, Section 13].

8.1.2. *Scheme Operation.* A shared secret key may be generated by each party by performing the following steps:

- (1) Establish the valid set of IQDL domain parameters with which the parties' key pairs shall be associated.
- (2) Establish a key pair $\mathcal{K}^{\text{IQDL}} = (\mathcal{K}_{\text{priv}}^{\text{IQDL}}, \mathcal{K}_{\text{pub}}^{\text{IQDL}}) = (a, \mathbf{a})$ for the operation, associated with the domain parameters established in step 1.
- (3) Obtain the other party's purported public key \mathbf{a}' for the operation, associated with the domain parameters established in step 1.
- (4) Compute a shared secret value \mathbf{c} from the private key a and the other party's public key \mathbf{a}' with the IQDL-SVDP-DH primitive.
- (5) Convert the shared secret value \mathbf{c} to an octet string using a suitable conversion primitive.
- (6) Derive a shared secret key from the octet string using the selected key derivation function.

9. INTELLECTUAL PROPERTY STATEMENT

The mere idea of using class groups (of imaginary quadratic and other number fields) is not patented. The algorithms for ideal arithmetic, including Rickert's and Schönhage's algorithms as well as NUCOMP and NUDUPL are not patented; no variants exist that are known to be patented. Specialised generic exponentiation algorithms may be patented.

IQDH is most likely covered by the Diffie-Hellman patent, which expired. IQ-Schnorr is potentially covered by the Schnorr patent; the variation due to Poupard and Stern is not known to be patented. IQ-GQ is most likely covered by the Guillou-Quisquater patent.

REFERENCES

- [1] Mark L. Bauer and Safuat Hamdy. On class group computations using the number field sieve (extended abstract). In Chi Sung Lai, editor, *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *LNCS*. Springer-Verlag, 2003. 311–325.
- [2] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
- [3] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures — how to sign with RSA and Rabin. In Ueli Maurer, editor, *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *LNCS*, pages 399–416. Springer-Verlag, 1996.
- [4] Johannes Buchmann. Algorithms for binary quadratic forms, 2002. Manuscript. <http://www.cdc.informatik.tu-darmstadt.de/~buchmann/AlgorithmsForQuadraticForms.ps>.
- [5] Henri Cohen. *A Course in Computational Algebraic Number Theory*, volume 138 of *GTM*. Springer-Verlag, 1995.
- [6] Niels Ferguson and Bruce Schneier. *Practical Cryptography*. John Wiley & Sons, 2003.
- [7] Louis C. Guillou and Jean-Jaques Quisquater. A practical zero-knowledge protocol fitted to security microprocessors minimizing both transmission and memory. In Christoph G. Günther, editor, *Advances in Cryptology – EUROCRYPT '88*, volume 330 of *LNCS*, pages 123–128. Springer-Verlag, 1988.

- [8] Safuat Hamdy. *Über die Sicherheit und Effizienz kryptographischer Verfahren mit Klassengruppen imaginär-quadratischer Zahlkörper*. PhD thesis, Technische Universität Darmstadt, Fachbereich Informatik, Darmstadt, Germany, 2002. <http://elib.tu-darmstadt.de/diss/000203>. In German.
- [9] Safuat Hamdy. IQ cryptography: A secure and efficient alternative. *Designs, Codes and Cryptography*, 2004. Submitted.
- [10] IEEE P1363. Ieee p1363: Standard specification for public-key cryptography. <http://grouper.ieee.org/groups/1363/>.
- [11] Michael J. Jacobson, Jr. *Subexponential Class Group Computation in Quadratic Orders*. PhD thesis, Technische Universität Darmstadt, Fachbereich Informatik, Darmstadt, Germany, 1999.
- [12] Michael J. Jacobson, Jr. and Alf van der Poorten. Computational aspects of NUCOMP. In Claus Fieker and David R. Kohel, editors, *Algorithmic Number Theory, ANTS-V*, volume 2369 of *LNCS*, pages 120–133. Springer-Verlag, 2002.
- [13] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. In Hideki Imai and Yuliang Zheng, editors, *Practice and Theory in Public Key Cryptography, PKC 2000*, volume 1751 of *LNCS*, pages 446–465. Springer-Verlag, 2000.
- [14] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
- [15] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
- [16] Guillaume Poupard and Jacques Stern. Security analysis of a practical “on the fly” authentication and signature generation. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT ’98*, volume 1403 of *LNCS*, pages 422–436. Springer-Verlag, 1998.
- [17] Neil W. Rickert. Efficient reduction of quadratic forms. In Erich Kaltofen and Stephen M. Watt, editors, *Computers and Mathematics, Cambridge, Massachusetts, 1989*, pages 135–139. Springer-Verlag, 1989.
- [18] Claus P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
- [19] Arnold Schönhage. Fast reduction and composition of binary quadratic forms. In Stephen M. Watt, editor, *Proc. ISSAC 91*, pages 128–133. ACM Press, 1991.
- [20] Alf van der Poorten. A note on NUCOMP. *Mathematics of Computation*, 72(244):1935–1946, 2003.

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF CALGARY, CALGARY, AB, T2N 1N4, CANADA
E-mail address: {hamdy, pauld}@math.ucalgary.ca