How (not) to hash into class groups of imaginary quadratic fields?

István András Seres*[1], Péter Burcsi†[1], and Péter Kutas‡ [1,2]

[1]Eötvös Loránd University
[2]University of Birmingham

May 9, 2024

Abstract

Class groups of imaginary quadratic fields (class groups for short) have seen a resurgence in cryptography as transparent groups of unknown order. They are a prime candidate for being a trustless alternative to RSA groups because class groups do not need a (distributed) trusted setup to sample a cryptographically secure group of unknown order. Class groups have recently found many applications in verifiable secret sharing, secure multiparty computation, transparent polynomial commitments, and perhaps most importantly, in time-based cryptography, i.e., verifiable delay functions, (homomorphic) time-lock puzzles, timed commitments, etc.

However, there are various roadblocks to making class groups widespread in practical cryptographic deployments. We initiate the rigorous study of hashing into class groups. Specifically, we want to sample a uniformly distributed group element in a class group such that nobody knows its discrete logarithm with respect to any public parameter. We point out several flawed algorithms in numerous publicly available class group libraries. We further illustrate the insecurity of these hash functions by showing concrete attacks against cryptographic protocols, i.e., verifiable delay functions, if they were deployed with one of those broken hash-to-class group functions. We propose two families of cryptographically secure hash functions into class groups. We implement these constructions and evaluate their performance. We release our implementation as an open-source library.

1 Introduction

Sampling a uniformly random group element is a fundamental cryptographic primitive in numerous protocols. It is applied in various algorithms to sample public parameters or map messages to group elements in encryption [BF01] or signature schemes [BLS01]. In some groups of interest, such as $\mathbb{Z}_p^*$ or $\mathbb{Z}_N^*$ (for a prime p and a semiprime N with unknown factorization) this is a simple task assuming a cryptographically secure hash function $H : \{0,1\}^* \to \{0,1\}^{2\lambda}$ for some security parameter λ, see efficient constructions in [KL07]. For other cryptographically interesting groups, such as elliptic curve groups, this is a non-trivial task that spurred a long line of research initiated mainly by Icart [Ica09] and refined by Farashahi, Shparlinski, and Voloch [FSV09] and extended to pairing-friendly curves (and is already being standardized [FHSS+23]) by Wahby and Boneh [WB19] among others [FFS+13, FT12, SBC+09].

Buchmann and Williams first proposed cryptography based on class groups of imaginary quadratic fields (class groups for short) in 1988 [BW88]. Recently, class groups have seen a revived interest in the cryptographic community as class groups offer a transparent instantiation for groups of unknown order. While it is cumbersome to establish an RSA group $\mathbb{Z}_N^*$ in a trust-minimized way, one either conducts a secure multi-party computation protocol among many peers to create a semiprime without unknown factorization [BF97, HMR+19, CHI+21, BDF+23] or samples a gigantic modulus that will be guaranteed to have large enough prime factors. On the other hand, class groups allow one to sample a group of unknown order transparently, i.e., by choosing the discriminant $\Delta = -p$ to be a large enough prime, as there is no efficient algorithm to compute the order of $\mathsf{Cl}(\Delta)$. Class groups were used to build numerous novel cryptographic protocols such as dynamic accumulators [Lip12], linearly homomorphic encryption scheme [CL15], polynomial commitments [BFS20a, AGL+22, SB23] and a plethora of protocols in time-based cryptography, for instance, verifiable delay functions [BBBF18, Pie19, Wes19], (homomorphic) time-lock puzzles [RSW96, MT19], timed commitments [BN00, TCLM21], etc. Even though class groups are a promising

*seresistvanandras@gmail.com.
†bupe@inf.elte.hu.
‡p.kutas@bham.ac.uk.

candidate to instantiate cryptographic protocols, they are both less studied from a theoretical (lack of more rigorous cryptanalysis [BKSW20]) and less developed from a practical (lack of more high-quality implementations [BCIL23] and developer tooling) point of view.

In this paper, we aim to bring class groups closer to real-world cryptographic deployments by initiating the study of cryptographically secure hash functions onto the elements of a class group of imaginary quadratic fields. To our knowledge, this is the first time one has built this fundamental building block for class groups of imaginary quadratic fields. The previous construction neither proves its security nor implements and evaluates its performance [Wes19]. Moreover, we improve and extend the aforementioned solution.

1.1 Applications of hashing into class groups

Hashing into reduced binary quadratic forms is essential in many cryptographic applications. We detail five protocols that rely on a cryptographically secure hash-to-class-group function.

Verifiable delay functions [BBBF18] In a verifiable delay function (VDF) $g : \{0,1\}^* \to \mathbb{G}$ for a group of unknown order $\mathbb{G}$ [RSS20], one computes $g(x) := H(x)^{2^T}$ for some time (or delay) parameter T, which is typically $T \approx 2^{35-45}$, and hash function $H(\cdot)$ that maps into $\mathbb{G}$ [Pie19, Wes19]. As Wesolowski noted in [Wes19], applying a hash function $H(\cdot)$ in the VDF $g(\cdot)$ is paramount. To see this, let us consider the variant of $g(\cdot)$, where there was no application of $H(\cdot)$ on the input x. Then observe that the adversary by computing $g(x) = x^{2^T} \in \mathbb{G}$ could obtain $g(x^\alpha)$ from $g(x)$ with a single exponentiation, that is, $g(x^\alpha) = g(x)^\alpha$, since $x \to x^{2^T}$ is a group homomorphism. This contradicts the sequentiality requirement of VDFs, as we require that $g(x)$ should take T sequential steps on *any input* x as opposed to the $\mathcal{O}(\log T)$ step to compute a single exponentiation.

At the time of this writing, the only deployed class group VDF is run by the Chia network [CP19]. They obviate the problem of hashing into a fixed class group by selecting a new class group $\mathsf{Cl}(\Delta)$ for each epoch by generating a random prime discriminant Δ such that $\Delta \equiv 7 \mod 8$. For every freshly generated $\mathsf{Cl}(\Delta)$, they start the VDF from the fixed form $(2, 1, \frac{\Delta+1}{8})$ which is a canonical ideal of norm 2. Interestingly, the isogeny-based VDF over $\mathbb{F}_p$ from [DFMPS19] is analogous to this construction in the following sense. Namely, computing powers of an ideal of norm 2 translates to a chain of 2-isogenies on the elliptic curve side. In [DFMPS19], they need a trusted setup to avoid certain shortcut attacks exactly because it is believed to be hard to hash onto the set of supersingular elliptic curves over $\mathbb{F}_p$ (or even to the full supersingular isogeny graph in general, for that matter [BBD+22]).

Contrary to the approach of Chia, we anticipate that future deployments of class group VDFs in programmable blockchains will likely use a fixed class group $\mathsf{Cl}(\Delta)$ for a *transparently and carefully* chosen Δ, i.e., such that it does not contain backdoors [BKSW20]. The use of a fixed, standardized class group is also motivated by prospective applications such as distributed randomness beacons [CMB23], time-stamping services [LSS20], consensus algorithms [Lon19], contract signing [BN00], or lotteries [GS98]. We envision that using a fixed class group in a programmable blockchain will ease these future applications' interoperability, programmability, and extendability.

Timed commitments from class groups [TCLM21] Thyagarajan et al. built an efficient CCA timed commitment in class groups. One of the key ingredients of their construction relies on sampling a well-formed, uniformly random public key $\mathsf{pk} \in \mathbb{G}$ that is a class group element. Their construction faces numerous technical challenges due to the lack of a secure hash function with the codomain of the applied class group $\mathbb{G}$. With such a cryptographic tool, their entire construction could be simplified and made more efficient.

Zero-knowledge proofs (of knowledge) in groups of unknown order [BBF19] Certain Σ-protocols in generic groups of unknown order can achieve at most soundness $1/2$ per challenge. This impossibility result was overcome in [BBF19], achieving negligible soundness error using a short unstructured common-reference string containing two fresh random generic group generators. In practice, one must sample these random group elements via a cryptographically secure hash-to-class group function.

Transparent polynomial commitment schemes [BFS20b, AGL+22, SB23] Recently, several transparent polynomial commitment schemes have been instantiated in class groups. These protocols often need an unstructured common reference string of many uniformly random class group elements. These group elements could be generated via a cryptographically secure hash-to-class group function.

The SPAR integer factoring algorithm [SL84] Finally, the integer factoring algorithm of Schnorr–Seysen–Lenstra also relies on the ability to sample uniformly random class group elements in $\mathsf{Cl}(\Delta)$.

1.2 Desiderata for a hash function into class groups

We call a hash function to a class group cryptographically secure if the function $\mathsf{H}_{\mathsf{BQForm}} : \{0,1\}^* \to \mathbb{G}$, maps to a class group $\mathbb{G}$, and satisfies the following four essential properties.

- **Deterministic, efficient, and constant time**. Evaluating the function $\mathsf{H}_{\mathsf{BQForm}}(\cdot)$ should be efficient and must take constant time on any input to avoid timing attacks. This is non-trivial already in the case of hashing to elliptic curve functions [Ica09].
- **Collision resistance**. It should be hard for any efficient adversary (i.e., probabilistic polynomial time (PPT)) to find two messages $m_1 \neq m_2$ such that $\mathsf{H}_{\mathsf{BQForm}}(m_1) = \mathsf{H}_{\mathsf{BQForm}}(m_2)$.
- **One-way function**. For any image $\mathsf{H}_{\mathsf{BQForm}}(m)$ of the hash function, no PPT adversary should be able to output the pre-image m with non-negligible probability $1/\mathsf{poly}(\lambda)$, where λ is the security parameter.
- **Uniformly distributed**. The hash function's $\mathsf{H}_{\mathsf{BQForm}}(\cdot)$ codomain must be uniformly distributed in the class group $\mathbb{G}$, i.e., among all reduced binary quadratic forms with a given discriminant Δ. This requirement eliminates many attempts implemented and used in various libraries implementing class group arithmetics.

Our contributions In this work, we make the following contributions.

Flawed hash functions. We review two hash functions onto class groups implemented in various open-source libraries, e.g., SageMath. We demonstrate that both fall short of being an efficient and secure hash function thanks to various vulnerabilities, e.g., the codomain of the hash function is nonuniform or highly skewed. Furthermore, we show concrete attacks against cryptographic protocols, i.e., verifiable delay functions, if they were deployed with one of these insecure hash-to-class group functions.

Provably secure hash constructions. We propose two families of secure and efficient hash functions with class group codomains. The first construction family is inspired by the key generation algorithm of CSIDH [CLM+18]. Meanwhile, the second family of hash functions extends and improves the hash function of Wesolowski [Wes19]. We prove both of our constructions' security by only relying on standard cryptographic assumptions, i.e., only assuming cryptographically secure hash functions to $\{0,1\}^\lambda$.

Open-source implementation. We build efficient algorithms that hash securely into class groups of imaginary quadratic fields. Additionally, we provide a publicly available implementation of our proposed hash functions in Python3 at the following link: `https://www.github.com/seresistvanandras/hashingToClassGroups`.

The rest of this paper is organized as follows. In Section 2, we provide the necessary background on (reduced) binary quadratic forms and class groups. In Section 3, we show several insecure but used methods to hash into class groups. In Section 4, we provide algorithms that are cryptographically secure hash functions and their codomain is a class group of imaginary quadratic fields. We detail our performance evaluation of the three proposed hash functions in Section 5. Finally, our paper concludes with open research problems and future directions in Section 6.

2 Preliminaries

2.1 Notations and definitions

The security parameter is denoted as λ. We assume a cryptographically secure hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{2\lambda}$ of λ bits of security. Furthermore, we also assume a deterministic hash function $\mathsf{H}_{\mathsf{Primes}} : \{0,1\}^* \to \mathsf{Primes}(\lambda)$ with codomain of prime numbers with λ bitlength. Such a primitive was first discussed and built by Cachin et al. in the context of private information retrieval [CMS99]. The divisor function $d(n) : \mathbb{N} \to \mathbb{N}$; $d(n) := \sum_{d|n} 1$ assigns the number of divisors to n. Let $\mathsf{factors}(n)$ denote the multiset of the prime factors of n, i.e., with multiplicity. Whenever we sample x from a set S uniformly at random, we write $x \in_R S$ or sometimes $x \xleftarrow{\$} S$. $U(c,d)$ denotes the uniform probability distribution on the $[c,d]$ interval. We denote by $b = \mathsf{SqrtModP}(\mathsf{a},\mathsf{p})$ the algorithm which returns b (the smallest positive) such that $b^2 \equiv a \mod p$ iff. $\left(\frac{a}{p}\right) = 1$, otherwise it returns $\perp$.

Most of the following definitions, lemmas, and theorems can be found in the excellent textbook of Buchmann and Vollmer on binary quadratic forms [BV07]. We refer to a binary quadratic form $f(x,y) = ax^2 + bxy + cy^2$ by the triple of its coefficients, i.e., $f = (a,b,c)$. In this work, we solely work with integral quadratic forms, i.e., $(a,b,c) \in \mathbb{Z}^3$.

Definition 1 (Discriminant of a form). *The discriminant Δ of a form $f = (a, b, c)$ is the integer $\Delta := b^2 - 4ac$.*

We work with (the equivalence classes of) forms with the same discriminant Δ in class group cryptography. Hence, when representing a form, we can usually drop the last coefficient c and refer to a form as the pair (a, b) since the fixed discriminant will be evident from the context. Note $\Delta \equiv 0 \bmod 4 \vee \Delta \equiv 1 \bmod 4$, though in our applications $\Delta < 0$ and prime. Sometimes we denote the discriminant of a form f as $\Delta(f)$. The matrix of a form is defined as $M(f) := \begin{pmatrix} \mathrm{a} & \mathrm{b}/2 \\ \mathrm{b}/2 & \mathrm{c} \end{pmatrix}$. Thus, the form $f(x, y)$ can be rewritten as $f(x, y) = (x, y)M(f)(x, y)^T$. We are interested in the transformations of a form f by linear transformations

$$U = \begin{pmatrix} \mathrm{s} & \mathrm{t} \\ \mathrm{u} & \mathrm{v} \end{pmatrix} \in \mathbb{Z}^{2\times 2}, \tag{1}$$

such that $det(U) = 1$, i.e., $U \in SL(2, \mathbb{Z})$. We define $U(x, y) := (sx + ty, ux + vy)$. Then $(fU)(x, y) = (det(U))f(U(x, y))$. We treat forms as equivalents that can be transformed into each other with linear transformations. An important infinite subgroup of $SL(2, \mathbb{Z})$ is generated by the matrix $T := \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ and we denote the generated subgroup as

$$\Gamma = \left\{ T^s = \begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix} : s \in \mathbb{Z} \right\}. \tag{2}$$

The cyclic group Γ acts on the set of positive definite forms. A small computation shows that the elements in the Γ-orbit of f can be given by

$$fT^s = (a, b + 2sa, f(s, 1)). \qquad (s \in \mathbb{Z}) \tag{3}$$

2.2 A primer on reduction theory and class groups

This paper focuses solely on forms with *negative discriminant* Δ, which are the most relevant to cryptography [BW88]. Moreover, we will only focus on *positive definite* binary quadratic forms as we define next.

Definition 2 (Positive definite forms). *A form f is positive definite if $\forall (x, y) \neq (0, 0), (x, y) \in \mathbb{R}^2 : f(x, y) > 0$.*

It is easy to show that a form $f = (a, b)$ is positive definite if and only if $\Delta(f) < 0 \wedge a > 0$.

Definition 3 (Normal binary quadratic form). *A form f is said to be* normal *if $-a < b \leq a$.*

Definition 4 (Reduced binary quadratic form). *We say that a positive definite form $f = (a, b, c)$ is reduced if it is* normal, *$a \leq c$, and if $b \geq 0$ for $a = c$.*

Given a non-reduced form, we can obtain a reduced form by the repeated application of the following two efficient operations.

Definition 5 (Normalization operation). *The normalization operation on the form (a,b,c) is given by $\eta(a, b, c) := (a, b + 2ra, ar^2 + br + c)$, where $r = \lfloor \frac{a-b}{2a} \rfloor$.*

Definition 6 (Reduction operation). *The reduction operation on the form (a, b, c) is defined as $\rho(a, b, c) := (c, -b + 2sc, cs^2 - bs + a)$, where $s = \lfloor \frac{c+b}{2c} \rfloor$. It also outputs a 2×2 matrix T such that $M(a, b, c)T = M(\rho(a, b, c))$.*

Unlike the normalization operation η, the reduction operation may need multiple iterations before reducing a form. To reduce a form f, one computes $f \leftarrow \eta(f)$ then repeatedly compute $f \leftarrow \rho(f)$ until f is reduced. It can be shown that the number of reduction steps performed by the reduction algorithm $\rho(\cdot)$ when applied to a positive definite form $f = (a, b, c)$ is at most $\lfloor \log_2(a/\sqrt{|\Delta|}) \rfloor + 2$.

It is easy to see that for a reduced binary quadratic form (a, b, c), we have that $|b| \leq a \leq \sqrt{\frac{|\Delta|}{3}}$. This is because we have then $|\Delta| = 4ac - b^2 \geq 4a^2 - a^2 = 3a^2$. A reduced form with a given discriminant Δ is typically represented as the pair (a, b), i.e., this representation has a size $\approx \log_2(|\Delta|)$. Dobson, Galbraith, and Smith showed how to compress a reduced form to a representation that only has $\frac{3}{4}\log_2(|\Delta|)$ bits [DGS20]. However, in this paper, we denote reduced binary quadratic forms simply with their uncompressed representation, i.e., (a, b). Sometimes, we explicitly spell out the full representation (a, b, c) of a (reduced) binary quadratic form. One can introduce an Abelian group operation and create a group of unknown order from (the equivalence classes of) reduced binary quadratic forms with a given discriminant Δ. This is the famous class group already known to Gauss in 1801. In the rest of this paper, we use the notations $\mathsf{Cl}(\Delta)$ and $\mathbb{G}$ interchangeably to mean a class group of the imaginary quadratic field with discriminant Δ. By a slight abuse of notation, we also denote the set of *reduced* binary quadratic forms with discriminant Δ by $\mathsf{Cl}(\Delta)$. Recall the class group $\mathsf{Cl}(\Delta)$ is isomorphic to the usual class group of fractional ideals modulo principal ideals [BV07].

Complexity of computing $|\mathsf{Cl}(\Delta)|$ There is no known efficient polynomial-time algorithm to compute the order of $\mathsf{Cl}(\Delta)$. Sutherland's algorithm has exponential worst-case complexity, i.e., $\mathcal{O}(\sqrt{|\mathsf{Cl}(\Delta)|/\log\log|\mathsf{Cl}(\Delta)|})$ [Sut07]. However, Sutherland's algorithm can exploit the structure of $\mathsf{Cl}(\Delta)$ and runs significantly faster when $|\mathsf{Cl}(\Delta)|$ is smooth. The security implications of Sutherland's algorithm in the context of class group cryptography are extensively studied in [DGS20]. Currently, the fastest algorithm to compute the order of the class group $\mathsf{Cl}(\Delta)$ runs in subexponential time that is $L_{|\Delta|}(1/2)$ and is due to Hafner and McCurley [HM89]. Moreover, Buchmann's algorithm can additionally compute the structure of $\mathsf{Cl}(\Delta)$. We refer the reader to Biasse et al. [BJJS10] for a modern treatment of the security of class group cryptography. We mention the current computational record for computing the order of a class group held by Beullens, Kleinjung, and Vercauteren, whereby the authors calculated $|\mathsf{Cl}(\Delta)|$ using the Hafner-McCurley algorithm for a discriminant Δ of 512-bits running in 52 core years [BKV19].

Definition 7 (Conductor)**.** *The conductor of a discriminant Δ is the largest positive integer f such that Δ/f^2 is a discriminant of a binary quadratic form. We denote it by $f(\Delta)$.*

A fundamental discriminant has $f(\Delta) = 1$. In our applications, we solely consider (negative) prime discriminants; hence, we only work with $f(\Delta) = 1$.

We denote the set of integral binary quadratic forms with a given discriminant and a coefficient as $\mathcal{F}(\Delta, a) := \{(a, b, c)\Gamma : b, c \in \mathbb{Z}, \Delta(a, b, c) = \Delta\}$. The number of such forms is denoted as $\mathcal{R}(\Delta, a) = |\mathcal{F}(\Delta, a)|$. Note the forms in $\mathcal{F}(\Delta, a)$ are not necessarily reduced, though by definition they belong to different Γ-orbits. It can be shown that the function $\mathcal{R}(\Delta, \cdot)$ is a multiplicative function, i.e., for coprime positive integers a_1, a_2 we have $\mathcal{R}(\Delta, a_1a_2) = \mathcal{R}(\Delta, a_1)\mathcal{R}(\Delta, a_2)$. Therefore, we need to establish the values of this function at prime powers p^e. Next, we recall this result in the following theorem whose proof can be found in [BV07].

Theorem 1 ($\mathcal{R}(\Delta, a)$ for a prime $a = p$ and prime power $a = p^e$.)**.** *Let p be a prime:*

- *If $a = p$, then $\mathcal{R}(\Delta, p) = \left(\frac{\Delta}{p}\right) + 1$.*
- *If $a = p^e$, and $e \geq 1$ then we have the following cases:*
  - *If $\left(\frac{\Delta}{p}\right) = -1$, then $\mathcal{R}(\Delta, p^e) = 0$.*
  - *If $\left(\frac{\Delta}{p}\right) = 0, e \geq 2, p \nmid f(\Delta)$, then $\mathcal{R}(\Delta, p^e) = 0$.*
  - *If $\left(\frac{\Delta}{p}\right) = 1$, then $\mathcal{R}(\Delta, p^e) = 2$.*

Theorem 2 (Establishing $\mathcal{F}(\Delta, p)$ for a prime p)**.** *If $\mathcal{R}(\Delta, p) \geq 1$, then we can compute the corresponding prime forms (p, b, c) as follows.*

- *If $p = 2$, if Δ is odd, then $b = 1$, else $b = 2(\Delta/4 \mod 2)$. Return: $(p, b, (b^2 - \Delta)/(4p))$.*
- *If $p \geq 3$, then let $b \leftarrow \mathsf{SqrtModP}(\Delta, p)$. Return: $(p, \pm b, (b^2 - \Delta)/(4p))$.*

Remark 1. *By multiplicativity of $\mathcal{R}(\Delta, \cdot)$ and by Theorem 1, $\mathcal{R}(\Delta, \cdot)$ is always a power of two when positive. Note if $a \leq \sqrt{\frac{|\Delta|}{4}}$, then all the forms in $\mathcal{R}(\Delta, a)$ are reduced since necessarily $|b| \leq a$, cf. Theorem 2, because $b^2 \leq a^2 \leq \frac{|\Delta|}{4}$. Therefore, the resulting form's c coefficient will be at least $c \geq \sqrt{\frac{|\Delta|}{4}}$, meaning that the form is reduced as $a \leq c$ holds as well. In other words, if $\sqrt{\frac{|\Delta|}{4}} \leq a \leq \sqrt{\frac{|\Delta|}{3}}$, then it might be the case that not all forms are reduced in $\mathcal{F}(\Delta, a)$. This phenomenon is clearly displayed in Figure 1a.*

Theorem 3 (Dirichlet's class number formula (1839))**.** *When $\Delta \leq -5$, Dirichlet's class number formula asserts that*

$$|\mathsf{Cl}(\Delta)| = \frac{\sqrt{|\Delta|}L(1, \chi_\Delta)}{\pi}. \tag{4}$$

Remark 2. *Typically $L(1, \chi_\Delta)$ is in the interval $[0.1, 10]$ [Sou07].*

3 How *not* to hash into class groups of imaginary quadratic fields?

This section reviews cryptographically insecure hash functions developed and used in various open-source libraries, e.g., the SageMath mathematical software library. We argue why these simple solutions do not meet our requirements and should not be used in security-critical applications. We denote the insecure hash functions in the sequel as $h^*(\cdot)$.

3.1 Group elements with known discrete logarithm

In some applications, it is necessary to sample group elements whose discrete logarithm is unknown to any other element from the set of public parameters. For instance, this is the case in the timed commitment application of class groups as proposed by [TCLM21]. Additionally, for certain zero-knowledge proofs of knowledge in groups of unknown order, if the adversary knows the discrete logarithm of the public parameters, they can break the soundness of the proof system [BBF19]. Therefore, in these applications, such a simple hash function as $h_1^* : \{0,1\}^* \to \mathbb{G} : h_1^*(m) := g^{h(m)}$ for a class group element $g \in \mathbb{G}$ and a cryptographically secure hash function h, is insecure.

3.2 Uniformly random b

A natural idea for choosing a random reduced quadratic form $(a,b,c) \in \mathsf{Cl}(\Delta)$ would be to first choose a uniformly random value $b \in [0, \sqrt{\frac{|\Delta|}{3}}]$ and specify a, c accordingly. Below, we investigate this approach by analyzing the distribution of coefficient b of reduced forms in $\mathsf{Cl}(\Delta)$. A similar strategy is employed by the most popular class group cryptography package on GitHub written in Rust[1]. This strategy's implicit assumption is that the distribution of the reduced form's b coefficient in $\mathsf{Cl}(\Delta)$ follows a discrete uniform distribution in $[-\sqrt{\frac{|\Delta|}{3}}, \sqrt{\frac{|\Delta|}{3}}]$. As we show in Theorem 4, this does not hold for random Δ: there is a bias in the distribution of b proportional to $\approx \frac{1}{2}\log\Delta - \log b$. This indicates that this natural idea fails to sample a uniform class group element from $\mathsf{Cl}(\Delta)$, hence contradicting our desiderata, cf. Section 1.2.

Definition 8 (The distribution of coefficient b in $\mathsf{Cl}(\Delta)$)**.** *Let us denote by $\mathcal{B}_\Delta$ the discrete probability distribution of the coefficient b of the reduced forms in $\mathsf{Cl}(\Delta)$, i.e.,*

$$\Pr[\mathcal{B}_\Delta = b] := \frac{|\{(a,c) | \exists (a,c) : (a,b,c) \in \mathsf{Cl}(\Delta)\}|}{|\mathsf{Cl}(\Delta)|}. \tag{5}$$

Note $\mathcal{B}_\Delta$ is a symmetrical distribution, i.e., $\forall b \in \mathbb{Z} : \Pr[\mathcal{B}_\Delta = b] = \Pr[\mathcal{B}_\Delta = -b]$ holds, since if $(a,b,c) \in \mathsf{Cl}(\Delta)$ then $(a,b,c)^{-1} = (a,-b,c) \in \mathsf{Cl}(\Delta)$. For an illustrative example of the probability distribution $\mathcal{B}_\Delta$, see Figure 1. Analogously, we define the discrete probability distribution of the coefficient a as follows.

Definition 9. *(The distribution of coefficient a in $\mathsf{Cl}(\Delta)$) Let us denote by $\mathcal{A}_\Delta$ the discrete probability distribution of the coefficient a of the reduced forms in $\mathsf{Cl}(\Delta)$, i.e.,*

$$\Pr[\mathcal{A}_\Delta = a] := \frac{|\{(b,c) | \exists (b,c) : (a,b,c) \in \mathsf{Cl}(\Delta)\}|}{|\mathsf{Cl}(\Delta)|}. \tag{6}$$

We compare the probability distributions of $\mathcal{A}_\Delta$ and $\mathcal{B}_\Delta$ to the discrete uniform distribution on the interval $I = [0, \sqrt{|\Delta|/3}]$ by applying Pearson's χ^2 statistic [Pea92]. Our null hypothesis is that $\mathcal{A}_\Delta$ and $\mathcal{B}_\Delta$ follow uniform distributions on I. We divide I into $\lceil \log(\sqrt{|\Delta|/3}) \rceil$ intervals and compute the corresponding χ^2 statistics. Given the extremely low p-values, we reject both hypotheses. For an illustrative example, see Table 1 for the chi-square statistics, and see Figure 1 for the actual empirical probability distributions for a medium-sized 30-bit discriminant.

Next, we characterize the asymptotic behavior of $\mathcal{B}_\Delta$, i.e., we describe the discrete probability distribution of the coefficient b of reduced forms in $\mathsf{Cl}(\Delta)$ when the discriminant Δ goes to infinity. For a medium-sized class group, see our measurement in Figure 1. Note that if $\mathcal{B}_\Delta$ would follow a uniform distribution, then we would have $\forall b \in [-\sqrt{|\Delta|/3}, \sqrt{|\Delta|/3}] : \Pr[\mathcal{B}_\Delta = b] = \frac{|\mathsf{Cl}(\Delta)|}{2\sqrt{\frac{|\Delta|}{3}}} \approx C_\Delta \left(= \frac{\sqrt{3}L(1,\chi_\Delta)}{2\pi}\right)$, where C_Δ is a constant dependent solely on the discriminant Δ, applying the Dirichlet class number formula, cf. Theorem 3.

We first give an estimate on the expected number of reduced forms where coefficient b is fixed, while the discriminant Δ is chosen uniformly from an interval.

[1] See ZenGo-X Class group implementation: https://github.com/ZenGo-X/class.

Coefficient a in $\mathsf{Cl}(-831,370,543)$										
a	$[0,l]$	$[l,2l]$	$[2l,3l]$	$[3l,4l]$	$[4l,5l]$	$[5l,6l]$	$[6l,7l]$	$[7l,8l]$	$[8l,9l]$	$[9l,10l]$
Uniform	1202.9	1202.9	1202.9	1202.9	1202.9	1202.9	1202.9	1202.9	1202.9	1202.9
Empirical	1345	1350	1288	1330	1274	1372	1324	1322	1174	250
Coefficient b in $\mathsf{Cl}(-831,370,543)$										
b	$[0,l]$	$[l,2l]$	$[2l,3l]$	$[3l,4l]$	$[4l,5l]$	$[5l,6l]$	$[6l,7l]$	$[7l,8l]$	$[8l,9l]$	$[9l,10l]$
Uniform	601.5	601.5	601.5	601.5	601.5	601.5	601.5	601.5	601.5	601.5
Empirical	2,148	1,100	816	655	452	343	252	144	72	33

Table 1: We compare the uniform distribution and the discrete probability distributions of the reduced forms' coefficients a and b, i.e., $\mathcal{A}_\Delta$ and $\mathcal{B}_\Delta$, for $\Delta = -831,370,543$, using Pearson's χ^2 test. We divided the $|\mathsf{Cl}(\Delta)| = 12,029$ class group elements into $\lceil \log(\sqrt{|\Delta|/3}) \rceil = 10$ intervals, i.e., $l = 1665$. In both cases, we observe extremely large χ^2 statistics, i.e., 861.73 and 6173.28 for coefficient a and b, respectively. The corresponding p-values are $1.08 \cdot 10^{-179}$ and 0. Hence, we reject both null hypotheses. Since $\mathcal{B}_\Delta$ is a symmetrical distribution, we only show here the positive part of the distribution. See also Figure 1 for a visualization of these discrete distributions.

Theorem 4 (Expected number of forms with given b)**.** *Let b be an arbitrary natural number and $D > b^2$. Choose Δ uniformly from the interval $[-2D,-D)$ with the condition that $b^2 \equiv \Delta \pmod 4$. Denote by $A_b(\Delta)$ the number of reduced forms in $\mathsf{Cl}(\Delta)$ with coefficient b. Then we have for the expected value of $A_b(\Delta)$:*

$$E_\Delta(A_b(\Delta)) = \frac{1}{2}\log D - \log b + \mathcal{O}(1), \tag{7}$$

where the $\mathcal{O}(1)$ is independent of b and D.

Proof. Recall that $A_b(\Delta)$ is equal to the number of integers a with $b \le a \le c$ where $c = \frac{b^2-\Delta}{4a} \in \mathbb{Z}$. The inequality $a \le c$ is equivalent to $a \le \sqrt{\frac{b^2-\Delta}{4}}$. Thus, we have to count the divisors of $\frac{b^2-\Delta}{4}$ that are between b and the square root. For an individual value of Δ, this is a hard computational task, but we can sum over all values of $\Delta \in [-2D,-D)$. We note that the values of $\frac{b^2-\Delta}{4}$ are exactly the integers in the interval $(\frac{b^2+D}{4}, \frac{b^2+2D}{4}]$. We give both lower and upper bounds, using the estimates $\frac{D}{4} < \frac{b^2-\Delta}{4} \le \frac{3D}{4}$.

$$\mathcal{C} := \sum_{\Delta\in[-2D,-D)} A_b(\Delta) = \sum_{x\in(\frac{b^2+D}{4},\frac{b^2+2D}{4}]} \sum_{a|x, b\le a\le\sqrt{x}} 1 \tag{8}$$

By changing the order of summations and using the lower bound $\frac{D}{4} < \frac{b^2-\Delta}{4}$, we obtain:

$$\mathcal{C} \ge \mathcal{C}' := \sum_{b\le a\le\sqrt{\frac{D}{4}}} \sum_{k, x=ak\in(\frac{b^2+D}{4},\frac{b^2+2D}{4}]} 1 \tag{9}$$

We can evaluate the inner sum by counting the multiples of $a \in [b, \frac{D}{4}]$:

$$\mathcal{C}' = \sum_{a=b}^{\sqrt{\frac{D}{4}}} \left(\left\lfloor \frac{b^2+2D}{4a} \right\rfloor - \left\lfloor \frac{b^2+D}{4a} \right\rfloor \right) \tag{10}$$

We omit the floor function $\lfloor(\cdot)\rfloor$ resulting in $\mathcal{O}(1)$ error in each term in the sum. Altogether, we have an error term in $\mathcal{O}(\sqrt{D})$. Now, we can move the terms independent from a out and get:

$$\mathcal{C}' = \left(\frac{b^2+2D}{4} - \frac{b^2+D}{4} \right) \sum_{a=b}^{\sqrt{\frac{D}{4}}} \frac{1}{a} + \mathcal{O}(\sqrt{D}) \tag{11}$$

By approximating the harmonic sum using Euler-MacLaurin formula in $[b, \sqrt{\frac{D}{4}}]$, we have:

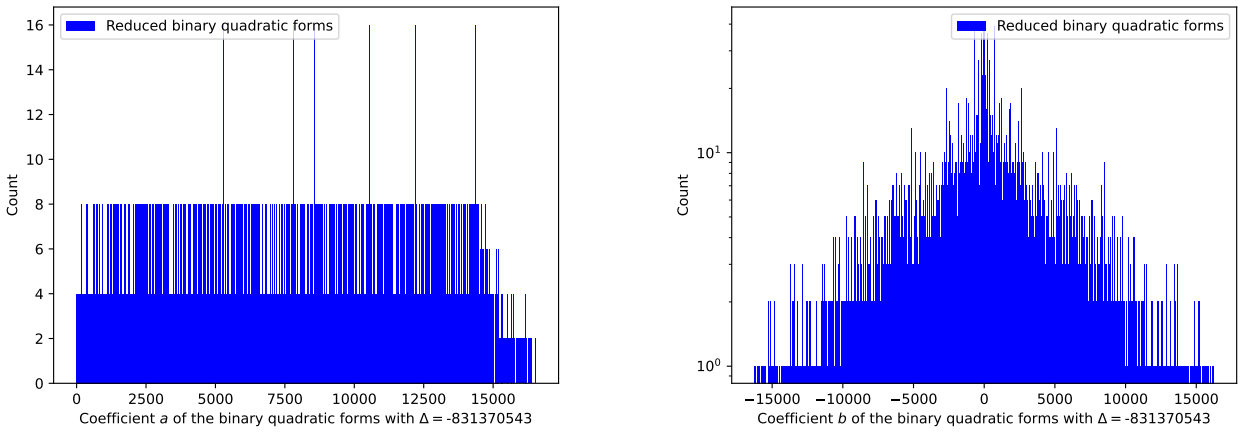
$$\mathcal{C}' = \frac{D}{4} \left(\log \sqrt{\frac{D}{4}} - \log b + \mathcal{O}(1) \right) + \mathcal{O}(\sqrt{D}) \tag{12}$$

The $\mathcal{O}(1)$ error term is multiplied by $\frac{D}{4}$, hence, it majorates the $\mathcal{O}(\sqrt{D})$ error term:

$$\mathcal{C}' = \frac{D}{4}\left(\frac{1}{2}\log D - \log b + \mathcal{O}(1)\right) \tag{13}$$

The upper bound derivation is similar. One only needs to modify $\mathcal{C}'$ in Equation (9) by letting the outer summation in $\mathcal{C}'$ run from b to $\sqrt{\frac{3D}{4}}$. The rest of the proof is identical.

Hence, the average value of A_b is $E(A_b) = \frac{\mathcal{C}}{D/4} = \frac{1}{2}\log D - \log b + \mathcal{O}(1)$. □



(a) Coefficient a.

(b) Coefficient b

Figure 1: Illustrating the distribution of the coefficient a and b of reduced binary quadratic forms in the class group $\mathsf{Cl}(\Delta)$ with the medium-sized discriminant $\Delta = -831,370,543$. The distribution of coefficient a is partially characterized in Theorem 1. Note the heights of the bars in the left figure are powers of two, except when $\sqrt{\frac{|\Delta|}{4}} \leq a \leq \sqrt{\frac{|\Delta|}{3}}$, i.e., towards the far right end of the figure, cf. Remark 1. We investigate the distribution of coefficient b in Theorem 4 and in Remark 3. Note the left figure is lin-lin, while the right is log-lin. We enclose an example distribution for coefficient c as well, cf. Figure 10.

Corollary 1. *Since the expected value of the uniform distributions behaves differently than the expected value of b established above, we conclude that sampling a uniformly random b (and choosing coefficient a accordingly) does not yield a secure hash-to-class group function.*

Remark 3. *Heuristically, it is natural to assume that the number and the sizes of divisors of $\frac{b^2-\Delta}{4}$ behave similarly to the statement of Theorem 4 whenever we consider a* fixed discriminant Δ *and let b chosen randomly from a sufficiently long interval, i.e., the length of the interval is in $\mathcal{O}(|\Delta|^{\varepsilon})$, where $0 < \varepsilon < \frac{1}{2}$. We leave it to future work to prove this heuristic statement precisely.*

Conjecture 1 (Distribution of b for a fixed Δ)**.** *Let $\Delta < 0$ be a* fixed discriminant*. Then there exist $c_1, c_2, \varepsilon > 0$ such that for a randomly chosen $b \in [B, 2B)$, with $B \in \mathcal{O}(|\Delta|^{\varepsilon})$, the average number of forms for b is:*

$$E_{b\in[B,2B)}(A_b(\Delta)) = c_1 \log|\Delta| - c_2 \log b + \mathcal{O}(1). \tag{14}$$

3.2.1 The discrete probability distribution of the coefficient c

Usually, the coefficient c is not part of the forms' representations as most libraries represent forms simply as (a, b). Consequently, we did not find any libraries that would have sampled a pseudorandom class group element by first sampling its c coefficient. Nonetheless, it might be the case that future hash-to-class group functions will use this direction. We leave this to future work. In Appendix A, we heuristically characterize the probability distribution of the c coefficients of reduced forms in $\mathsf{Cl}(\Delta)$. Additionally, we provide the empirical distribution of the c coefficients for a medium-sized discriminant, cf. Figure 10.

How *not* to hash into class groups? vol. #3.

Public parameters: $\mathbb{G} \xleftarrow{R} GGen(\lambda), \mathsf{H} : \{0,1\}^* \to \{0,1\}^{2\lambda}$ a cryptographically secure hash function.
Public inputs: $x \in \{0,1\}^*$.

1. $cnt = 0$.
2. $h \leftarrow \mathsf{H}(x||cnt)$. Let $(a,b) := (h[0:\lambda], h[\lambda:2\lambda])$. //Note the form (a,b) is most likely a non-reduced form.
3. If $(a,b) \in \mathsf{Cl}(\Delta)$ then: Return $\rho(a,b)$.
   - else $cnt = cnt + 1$, and goto 2.

Figure 2: This hash-to-class group function is implemented and used by the SageMath open-source mathematical software system. The application of this hash function in certain cryptographic protocols, e.g., in verifiable delay functions, makes the cryptosystem's deployment completely insecure, cf. Section 3.4.

3.3 Random form and reduce

Another seemingly correct approach might be to sample uniformly random a binary quadratic form $(a,b) \xleftarrow{R} [0, \sqrt{\frac{|\Delta|}{3}}]^2$ or from some other simple interval $[a,b]^2$. This approach is implemented by the Sage Quadratic Forms package[2]. It is well documented [3]. The problem of these approaches is that the resulting *reduced* binary quadratic forms, i.e., the class group elements, are not uniformly distributed in those intervals. In particular, as we show next, this would result in a highly skewed non-uniform distribution of reduced binary quadratic forms. Therefore, this approach is also completely insecure for cryptographic purposes, cf. Section 3.4. The next theorem formally characterizes the probability distribution of class group elements output by the function above. For an empirical measurement, see Figure 3a.

Theorem 5 (Distribution of random forms)**.** *Let $h_3^*(\cdot) : \{0,1\}^* \to \mathbb{G}$ denote the insecure hash function described above, cf. Figure 2. The output distribution of $h_3^*(\cdot)$ follows a power-law distribution in the class group $\mathbb{G}$. More formally, $\forall a \in [1, \sqrt{|\Delta|/3}]$ we have*

$$\Pr_s[h_3^*(s) = (a',b') | (a,b) \in \mathsf{Cl}(\Delta) \wedge \rho(a',b') = (a,b)] = \mathcal{O}\left(\frac{1}{a}\right). \tag{15}$$

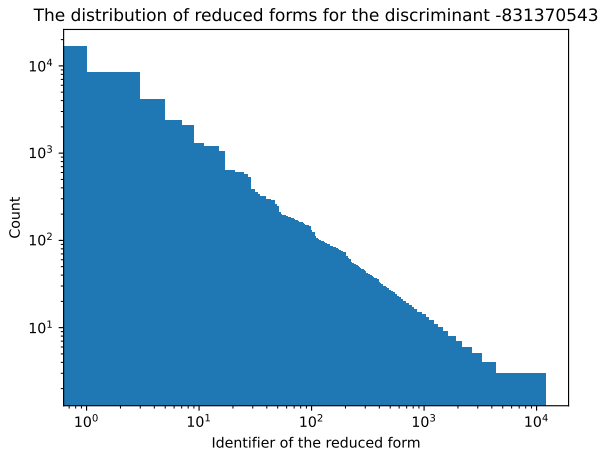
Proof. Let $f = (a,b) \in \mathsf{Cl}(\Delta)$ be a reduced binary quadratic form, and we assume that f was sampled from $[0, \sqrt{\frac{|\Delta|}{3}}]^2$. An easy calculation shows that the Γ-orbit of f contains forms $fT^s = (a, b+2sa, f(s,1))$ for $s \in \mathbb{Z}$, cf. Equation (3). We observe that the generated Γ-orbit's cardinality in the considered interval $[0, \sqrt{\frac{|\Delta|}{3}}]^2$ is $\mathcal{R}_{f\Gamma} := \left|\left\{fT^s | s \in \mathbb{Z} \wedge fT^s \in [0, \sqrt{\frac{|\Delta|}{3}}]^2\right\}\right| = \left\lfloor \frac{\sqrt{\frac{|\Delta|}{3}} - b}{2a} \right\rfloor$. Denoting the set of all quadratic forms in $[0, \sqrt{\frac{|\Delta|}{3}}]^2$ with discriminant Δ by $\mathsf{A}(\Delta)$, we get the desired $\frac{\mathcal{R}_{f\Gamma}}{|\mathsf{A}(\Delta)|} = \mathcal{O}(\frac{1}{a})$ probability stated in Equation (15). We remark that it is enough to consider only the Γ-orbit of a form f (the subgroup of $SL(2,\mathbb{Z})$ generated by the matrix T, cf. Equation (3)), as transformations by other matrices would result in forms f' such that $f' = (a,b) \notin [0, \sqrt{\frac{|\Delta|}{3}}]^2$. □

3.4 Breaking cryptographic protocols instantiated with flawed hash-to-class group functions

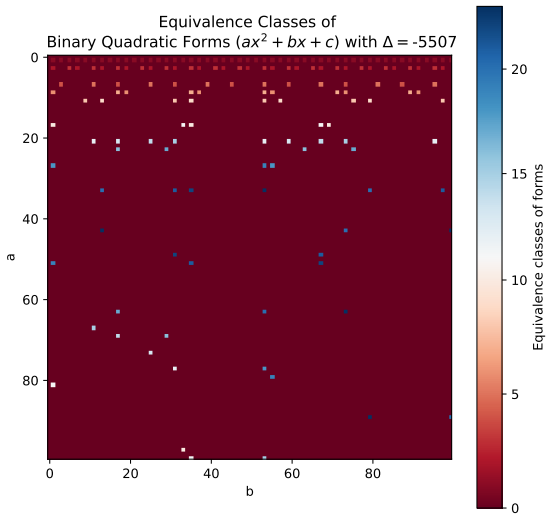
Next, we show that certain cryptographic protocols become completely insecure when instantiated with one of the insecure hash-to-class group functions described in this section. As an illustrative example, we consider a verifiable

[2]See SageMath random quadratic form function implementation: https://github.com/sagemath/sage/blob/develop/src/sage/quadratic_forms/random_quadraticform.py.

[3]See SageMath random quadratic form function documentation: https://doc.sagemath.org/html/en/reference/quadratic_forms/sage/quadratic_forms/random_quadraticform.html.



(a) The skewed power-law distribution of reduced binary quadratic forms among all forms in $[0, \sqrt{|\Delta|/3}]^2$ for the medium-sized discriminant $\Delta = -831,370,543$.



(b) The distribution of equivalent binary quadratic forms in $[0, \sqrt{|\Delta|/3}]^2$ for a small-sized discriminant $\Delta = -5,507$.

Figure 3: A family of insecure hash functions chooses uniformly random forms (a, b) from the square $[0, \sqrt{|\Delta|/3}]^2$ and subsequently reduces it. In these two figures, we analyze the distribution of the output of this hashing strategy induced on the reduced forms, i.e., group elements in $\mathsf{Cl}(\Delta)$. The deployment of this broken hash function would lead to attacks in certain cryptographic applications, e.g., verifiable delay functions, cf. Section 3.4.

delay function (VDF) protocol instantiated with the hash function described in Section 3.3. More formally, we consider the VDF function $g^*(x) : \{0,1\}^* \to \mathbb{G}; x \to (h_3^*(x))^{2^T} \in \mathbb{G}$, where T is the delay parameter and $h_3^*(\cdot)$ is the insecure hash function presented in Section 3.3. We show that an adversary who precomputes all the VDF values z^{2^T} for $\forall z : z = (a, b) \in \mathsf{Cl}(\Delta) \wedge a \in [0, \mathsf{poly}(\lambda)]$ will be able to compute the VDF output with non-negligible probability in λ without computing T sequential squarings in $\mathbb{G}$. In other words, a VDF adversary can exploit the skewed power law distribution of the insecure hash function $h_3^*(\cdot)$ on the class group $\mathsf{Cl}(\Delta)$ by anticipating that the hash output will likely produce forms (a, b) with small norms $a \in [0, \mathsf{poly}(\lambda)]$.

We say that a precomputing adversary $\mathcal{A}_{g^*}^{h_3^*}(\lambda)$ outputs 1 and breaks the VDF $g^*(\cdot)$ iff. $h_3^*(s) = (a, b)$ such that $a \leq \mathsf{poly}(\lambda)$. Next, we compute the probability that the adversary breaks the (sequentiality of the) VDF. Let $\mathcal{B} \in \mathsf{poly}(\lambda)$.

$$\Pr[\mathcal{A}_{g^*}^{h_3^*}(\lambda) = 1] \approx \frac{\int_1^{\mathcal{B}} \frac{1}{x} dx}{\int_1^{\sqrt{|\Delta|/3}} \frac{1}{x} dx} = \frac{[\ln x]_1^{\mathcal{B}}}{[\ln x]_1^{\sqrt{|\Delta|/3}}} = \frac{\ln(\mathcal{B})}{\ln(\sqrt{|\Delta|/3})} = \log_{\sqrt{|\Delta|/3}}(\mathcal{B}). \tag{16}$$

By choosing $\mathcal{B} := \sqrt{|\Delta|/3}^{1/\lambda}$, we ensure that $\Pr[\mathcal{A}_{g^*}^{h_3^*}(\lambda) = 1] \approx \frac{1}{\lambda}$ that is non-negligible in λ, while $\mathcal{B} \in \mathsf{poly}(\lambda)$. Let us consider the following concrete parameters. For $\lambda = 85$-bit security, we need a discriminant Δ with 3400 bits [DGS20]. Thus, if one chooses $\mathcal{B} := \sqrt{|\Delta|/3}^{1/\lambda} \approx 2^{20}$, then they need to precompute roughly $\mathcal{B}$ VDF evaluations on binary quadratic forms (a, b) such that $a \in \mathcal{B}$. This computation could be done in parallel time T using $\mathcal{B}$ processors. The success probability of this preprocessing attack is roughly $\approx \frac{20}{1700} = 1.17\%$, i.e., non-negligible in λ.

4 How to hash into class groups of imaginary quadratic fields?

In this section, we show how to hash securely and efficiently into class groups. Additionally, we prove the security of our constructions, investigate their friendliness to zero-knowledge proofs, and secure multiparty computation applications.

How to hash into class groups? Construction. #1.

Public parameters: $\mathbb{G} \xleftarrow{R} GGen(\lambda), \mathsf{H} : \{0,1\}^* \to \{0,1\}^d$ a secure hash function. A precomputation outputs $\{f_i\}_{i=1}^d \in \mathsf{Cl}(\Delta)^d$ forms with small prime norms, i.e., $\forall i \in [1,d] : f_i = (p_i, b_i) \in \mathsf{Cl}(\Delta)$ for some "small" prime $p_i \in [2, \mathcal{O}(\log|\Delta|)]$.
Inputs: $x \in \{0,1\}^*$.

1. Let $b := \mathsf{H}(x)$ and b_i be the ith bit of b.
2. Return: $\prod_{i=1}^{d} f_i^{b_i} (\in \mathsf{Cl}(\Delta))$.

Figure 4: This hash function to class groups of imaginary quadratic fields is reminiscent of the key generation algorithm of CSIDH [CLM+18]. The number of the generating ideals d to achieve 2^{-k} statistical uniformity of the codomain is precisely defined in Theorem 6.

4.1 Construction #1: Using a generating set of ideals à la CSIDH.

The idea of the following construction is reminiscent of the secret key generation algorithm of CSIDH [CLM+18]. In a pre-computation phase, we sample $f_1, \ldots, f_d$ binary quadratic forms with small prime norms (the a coefficient of the form is a small prime $p \in \mathcal{O}(\log|\Delta|)$) in $\mathsf{Cl}(\Delta)$). Heuristically, these group elements are well-distributed and approximately generate the whole group $\mathsf{Cl}(\Delta)$ if the number of generating ideals d is chosen properly [Dix08]. We hash the input x with the hash function $\mathsf{H}(\cdot)$ to obtain random bits $b = (b_1, \ldots, b_d)$ and output the approximately equidistributed group element $\prod_{i=1}^{d} f_i^{b_i}$. Next, we prove the security of this hash construction.

Theorem 6 (Construction #1. is secure)**.** *The hash function described in Section 4.1 is a cryptographically secure hash function with the codomain of a class group* $\mathsf{Cl}(\Delta)$ *assuming a one-way function* $\mathsf{H}(\cdot)$ *onto* $\{0,1\}^d$ *whenever* $d \geq 2\log|\mathsf{Cl}(\Delta)| + h + 2k$ *yielding a* 2^{-k}*-uniform distribution in* $\mathsf{Cl}(\Delta)$ *with probability at least* $1 - 2^{-h}$*.*

Proof. We show that Construction #1. satisfies our desiderata. We will first handle the one-way and the collision resistance properties together. The output of the function is an ideal whose prime factors lie in the set $f_1, \ldots, f_d$. Thus, by simple trial division, one can factor this ideal and retrieve the output of the hash function H. Thus, if one could find a collision or invert this function, that would imply that H is not cryptographically secure (i.e., one-way), which is a contradiction.

Now we look at the uniform distribution property. Since the construction has the same image as the Naor-Reingold type PRF from [BKW20, Theorem 23], the same theorem applies in this context as well. Similar constructions also appear in [MOT20] and [ADFMP20]. The group theoretic statement with regards to the codomain's statistical uniformity was proved in [Dix08, Theorem 3.]. □

4.2 Construction #2: Wesolowski's construction.

A folklore approach was proposed by Wesolowski in the conference version's appendix of [Wes19] and implemented by Pope [4]. The construction's idea is to generate a uniformly random prime $p \in [0, \sqrt{\frac{|\Delta|}{4}}]$, and if $\left(\frac{\Delta}{p}\right) = 1$ (which should happen on average with probability $\approx 1/2$), we can choose from two binary quadratic forms (p, b) and $(p, -b)$, where b is obtained by a modular square root computation, i.e., $b := (\sqrt{\Delta} \bmod p)$.

Theorem 7 (Construction #2. is secure)**.** *The hash function in Figure 5 is a cryptographically secure hash function with the codomain of a class group of imaginary quadratic fields assuming that a* $\mathsf{H}_{\mathsf{Primes}}(\cdot)$ *hash function with prime number codomain exists.*

Proof. We show that Construction #2. satisfies our desiderata. It is easy to see that all the desired properties (i.e., collision resistance, one-wayness, and uniform distribution of the codomain) can be reduced in a straightforward manner to the corresponding property of the underlying hash function $\mathsf{H}_{\mathsf{Primes}}(\cdot)$. □

[4] See: https://github.com/GiacomoPope/ClassGroups/blob/main/classgroup.py.

How to hash into class groups? Construction. #2.

Public parameters: $\mathbb{G} \xleftarrow{R} GGen(\lambda), \mathsf{H}_{\mathsf{Primes}} : \{0,1\}^* \to \mathsf{Primes}(\lambda)$ a cryptographically secure hash function.
Inputs: $x \in \{0,1\}^*$.

1. $i = 0$.
2. Compute $p := \mathsf{H}_{\mathsf{Primes}}(x||i)$. // For the sake of uniform distribution p must be large, i.e., $p \xleftarrow{\$} [0, \sqrt{|\Delta|/4}]$. We also need to ensure the reducedness of the form.
3. If $\left(\frac{\Delta}{p}\right) = 1$ then let $b := (\sqrt{\Delta} \bmod p)$.
   - else $i = i + 1$, and *goto* 2. //There is no prime form $(p, \cdot) \in \mathsf{Cl}(\Delta)$, cf. Theorem 1.
4. Let $s := (p \equiv 3)$. // If $\left(\frac{\Delta}{p}\right) = 1$, then there are two modular square roots: b and $-b$. We need to choose one of them deterministically.
   - If $s == 0 \vee s == 1$: Return (p, b).
   - elif $s == 2$: Return $(p, -b)$.

Figure 5: Wesolowski's original hash-to-class group construction had been implicitly introduced (though without security proofs, implementation, and performance evaluation) in the conference version of [Wes19]. We extend and improve this construction in Figure 6.

Remark 4. *One small aspect of both constructions is that collisions can occur at places that are not collisions for the utilized cryptographically secure hash function, as two different ideals can represent the same element in the class group. However, these are usually pretty rare, and if a relatively small number of these collisions are found, then there is a simple way of computing the order of the class group. Let $I_1, \ldots, I_n$ be a fixed set of prime ideals. Integer vectors of the form $(k_1, \ldots, k_n)$ for which $\prod_{i=1}^{n} I_i^{k_i}$ is principal clearly form a lattice (this is often called the relation lattice). If one can find n linearly independent vectors in this lattice (thus a lattice basis), then one can compute the determinant of the lattice in polynomial time, which is known to be equal to the class number. For further details, the reader is referred to [BKV19].*

4.2.1 Improving Wesolowski's construction

We want to highlight two major rooms for improvement in the hash function's original version as proposed in the appendix of the conference version of [Wes19], cf. Figure 5.

- **Inefficiency.** Wesolowski's construction is rather inefficient compared to the first proposed construction, as we show empirically in Section 5.2. This is due to the costly primality checks in the second step of the algorithm. Another computational bottleneck is the modular square root function calculation in the third step, as it must be computed modulo a large prime p, cf. Section 5.1.
- **Lack of surjectivity.** Even though the codomain of this hash function is uniformly distributed in $\mathsf{Cl}(\Delta)$, it might be desirable to produce not only prime forms as the output of a hash-to-class group function.

We can kill these two birds with one stone. The high-level idea of our improved version of Wesolowski's construction in Figure 6 is as follows. Instead of just generating prime forms $(p, b) \in \mathsf{Cl}(\Delta)$, for some large prime p, we will allow the first step of the algorithm to sample any composite integer $a \in_R [0, \sqrt{|\Delta|/3}]$. However, we need to sample carefully the coefficient a, as we need its factorization as well to be able to compute the corresponding coefficient b. For computing the coefficient b, we must be able to compute modular square roots modulo a as $b = \sqrt{\Delta} \mod a$, cf. Theorem 2. However, if one does not know the factorization of a, then this is a hard problem, also known as the RSA assumption. In fact, for a semiprime N, the ability to compute modular square roots is equivalent to factoring. Therefore, we need to sample a random coefficient a along with its factorization. Efficient algorithms to generate a random factored number are available due to Bach [Bac88]. We recall that in our specific

How to hash into class groups? Construction. #2. (Improved version)

Public parameters: $\mathbb{G} \xleftarrow{R} GGen(\lambda), \mathsf{H}_{\mathsf{Primes}} : \{0,1\}^* \to \mathsf{Primes}(\lambda)$ a cryptographically secure hash function.
Inputs: $x \in \mathbb{Z}, \mathsf{Cl}(\Delta), N$.

1. $(a, \mathcal{P}) \xleftarrow{\$}$ **BachWithBias**(N, Δ).// $\mathcal{P} = \mathsf{factors}(a)$. See Figure 7.
2. Apply Chinese-Remainder Theorem to compute $[B] := (\sqrt{\Delta} \bmod a)$. // We know the factors of a. Note $[B]$ is a list.
3. Let δ denote the number of prime factors of a. Note B has 2^δ elements. Let $b := B[a \mod 2^\delta]$. *Return:* (a, b). //We deterministically select one of the modular square roots using the previously generated factored random a.

Figure 6: An improved version of Wesolowski's hash function.

application, we need to generate these factored random numbers with a bias toward integers having multiple prime factors. This is due to the theorem that characterizes the coefficient a in reduced forms, cf. Theorem 1. Therefore, we slightly need to modify Bach's algorithm, cf. Figure 7.

Generating factored random numbers [Bac88] with
prime factor multiplicity bias

Public parameters: Δ discriminant, $N_0 \in \mathbb{Z}^+$ small integer, $y = 1$, $\mathcal{P} = \{\emptyset\}$ the multiset of prime factors.
BachWithBias(N, Δ):

1. If $N \leq N_0$. Sample $x \in_R (N/2, N]$ with bias $\mathcal{R}(\Delta, a)$. Factor x. *Return:* $(x, \mathsf{factors}(x))$.
2. While: True.
   (a) $q, \mathcal{P} :=$ genFactorWithBias(N,Δ) // $q = p^\alpha \wedge 2 \leq q \leq N$.
   (b) $N' := N/q$.
   (c) $(z, \mathcal{P}') \xleftarrow{\$}$**BachWithBias**$(N', \Delta)$.
   (d) $y := q \cdot z, \mathcal{P} := \mathcal{P} \cup \mathcal{P}'$. Sample $\mu \in_R U(0,1)$.
   (e) if $\gcd(q, z) = 1$, then $\mu := \mu / \mathcal{R}(\Delta, q)$.
   (f) If $\mu \leq \frac{\log N/2}{\log y}$ then *Return:* $(y, \mathcal{P})$.

Subroutine: genFactorWithBias(N,Δ):

1. Sample an integer $j \in_R [1, \log_2(N)]$. Let $q := 2^j + r$, where the integer r is such that $r \in_R [0, 2^j]$.
2. Sample $\nu \in_R U(0,1)$.
   (a) If q is a prime power $q = p^\alpha \wedge q \leq N \wedge \nu \leq \delta_N(q) \cdot 2^{\log_2(q)} \wedge \left(\frac{\Delta}{p}\right) = 1$ then *Return:* $(q, \mathsf{factors}(q))$. //We define $\delta_N(q)$ in Section 4.2.1.
   (b) else goto 1.

Figure 7: A modified version of Bach's factored random number generation algorithm [Bac88]. In our hash-to-class group application, we need to sample factored random numbers proportional to 2^k, where k is the number of their different prime factors. Moreover, any prime factor p must also satisfy $\left(\frac{\Delta}{p}\right) = 1$. Modifications are written in red.

We remark that in Step (2a) in Figure 7 of the genFactorWithBias(N, Δ) subroutine, one can efficiently decide

whether q is a prime power. Generating random numbers with the appropriate bias is achieved through the adjustment in Step (2e) $\mu := \mu/\mathcal{R}(\Delta, q)$ which multiplies the overall bias of the returned x by $\mathcal{R}(\Delta, q)$ for each prime power factor q, resulting in the correct bias by the multiplicative property of $\mathcal{R}(\Delta, \cdot)$, cf. Theorem 1. The function $\delta_N(q)$ for a prime power $q = p^\alpha$ is defined as $\frac{\log p}{\log N} \cdot \frac{\mathrm{RF}(N/2q, N/q)}{N}$, where $\mathrm{RF}(K, L)$ is the number of integers x in the interval $(K, L]$ counted with multiplicity $\mathcal{R}(\Delta, x)$. The proof that the generated random numbers have the desired distribution is essentially identical to that of Theorem 1 in [Bac88]. For efficiency, $\mathrm{RF}(\cdot, \cdot)$ should be approximated for large N. Heuristically, a number a with k distinct prime factors has a $1/2^k$ probability of having positive bias $\mathcal{R}(\Delta, x)$, and then the bias is 2^k (with slight adjustments when 2 is one of the prime factors). Therefore, $\mathrm{RF}(K, L)$ is expected to be around $L - K$. Based on empirical observations, we recommend using the approximation $\mathrm{RF}(K, L) \approx c_\Delta \cdot (L - K)$, where c_Δ can be estimated by looking at values of L and K where an exact computation of $\mathrm{RF}(K, L)$ is feasible.

Wesolowski's extended hash-to-class group function is enclosed in Figure 6 along with its subroutine, our modified Bach's algorithm in Figure 7 that samples a uniform reduced binary quadratic form in $\mathsf{Cl}(\Delta)$. We evaluate its improved performance in Section 5.2.

4.3 Zero-knowledge and secure Multiparty computation (MPC) friendliness

A recent line of research in symmetric-key primitives investigates the possibility of constructing hash functions, pseudorandom functions that are efficient both in zero-knowledge proofs and multiparty computation (MPC) applications [GRR+16], i.e., they have minimal multiplicative complexity [AGR+16]. The MiMC, Poseidon, and Reinforced concrete hash functions [GKR+21, GKL+22] are notable constructions, among many others.

Our proposed hash functions do not have minimal multiplicative complexity. Hence, verifying the hash function evaluation (i.e., proving that $y = \mathsf{H}(x)$ in zero knowledge for a secret witness x) in an arithmetic circuit will be somewhat computationally intensive. However, we describe a few simple tricks that can make the implementation of our hash-to-class group functions significantly more efficient in zkSNARK and MPC applications. A recurring observation in designing circuit-friendly algorithms is to realize that for most functions, e.g., division ($y \stackrel{?}{=} \frac{x}{a}$), it is significantly faster to verify the correctness of the function in the circuit (e.g., by checking that $y \cdot a \stackrel{?}{=} x$) than evaluating the function in the "forward" direction [KPS18]. We apply this insight in the class group context.

Reduction of binary quadratic forms. One can verify the computational integrity of the reduction of a form f to $\rho(f)$, cf. Definition 6. Instead of directly computing the reduction function $\rho(\cdot)$ in the arithmetic circuit, the prover can additionally supply the reduction matrix T, and the circuit would only verify a matrix multiplication, i.e., $M(f) \cdot T \stackrel{?}{=} M(\rho(f))$, where $M(\cdot)$ is the matrix form of a binary quadratic form, cf. Section 2.1. Additionally, the circuit must check that $\rho(f) = (a, b)$ is a reduced binary quadratic form, i.e., $|b| \leq a$, and that $T \in SL(2, \mathbb{Z})$ by confirming that $det(T) = 1$. Analogously, one can verify efficiently that two forms f, g are equivalent without the costly reductions in the circuit. Specifically, the prover needs to provide $T \in SL(2, \mathbb{Z})$ such that $M(f) \cdot T \stackrel{?}{=} M(g)$, which can be verified efficiently inside the circuit.

Class Group operation. The class group operation is notoriously slow, e.g., compared to the elliptic curve group operation. Our first proposed hash function evaluates $\mathcal{O}(\log|\Delta|)$ class group operations. Directly computing these class group operations is prohibitively expensive as each class group requires $\mathcal{O}(\log|\Delta|)$ reduction steps. Thus, the total number of reduction steps would be $\mathcal{O}(\log^2|\Delta|)$ for a single hash evaluation, which is costly. However, instead of directly computing $f_1 \cdot f_2 \in \mathbb{G}$ in the arithmetic circuit, we can more efficiently verify the correctness of $f_1 \cdot f_2 = f_3$ for any $f_1, f_2, f_3 \in \mathsf{Cl}(\Delta)$. Specifically, this could be achieved as follows. Let $f_1 = ax_1^2 + bx_1y_1 + cy_1^2$ and $f_2 = \alpha x_2^2 + \beta x_2y_2 + \gamma y_2^2$ be the operands. Note, that we consider (x_1, y_1) and (x_2, y_2) as independent variables. Thus, the prover's goal is to prove the correctness of a polynomial multiplication over $\mathbb{Z}$ such that $f_1 \cdot f_2 = f_3$ for $f_3 = AX^2 + BXY + CY^2$, where $X = jx_1x_2 + kx_1y_2 + ly_1x_2 + my_1y_2$ and $Y = rx_1x_2 + sx_1y_2 + ty_1x_2 + uy_1y_2$. The prover sends $A, B, C, X(= (j, k, l, m)), Y(= (r, s, t, u))$ of f_3 to the verifier. Afterwards, the verifier samples a uniform $r \in_R [0, 2^\lambda]$ and checks that $f_1(r) \cdot f_2(r) \stackrel{?}{=} f_3(r)$. We can make this proof system non-interactive using the standard Fiat-Shamir transformation [FS86]. In this protocol's non-interactive version, the verifier's challenge r is obtained as $r \leftarrow \mathsf{H}(f_1, f_2, A, B, C, X, Y)$ for a cryptographically secure hash function H to $\{0,1\}^{2\lambda}$. The soundness of the proof system is guaranteed by applying the Schwartz–Zippel lemma for polynomial identity testing.

Squaring. In a similar manner, one can efficiently verify a squaring of a form $f = (a, b, c)$ in $\mathsf{Cl}(\Delta)$, i.e., $f \cdot f \stackrel{?}{=} g$. The result f^2 should be $f^2 = (a^2, b - 2a\mu, \mu^2 - \frac{b\mu - c}{a})$, where μ is the solution of the linear congruence $bx \equiv c$

mod a. One needs to prove in the circuit that for μ, it indeed holds $b\mu \equiv c \mod a$. Once μ is verified, f^2 can be computed by the arithmetic circuit. Verifying μ is done efficiently if the prover additionally discloses $l \in \mathbb{Z}$ and the circuit checks that $b\mu - c = la$. Therefore, verifying one squaring operation requires four integer multiplications to establish (a, b) inside the arithmetic circuit.

5 Implementation and Performance Evaluation

In this section, we report our open-source implementation and evaluate the performance of the cryptographically secure hash function constructions described in Section 4.

5.1 Theoretical Performance Evaluation

Next, we evaluate the computational complexity of the suggested hash functions.

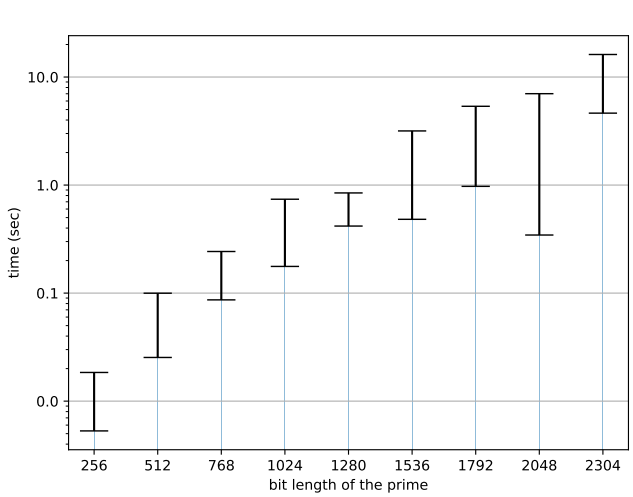
CSIDH hash (cf. Section 4.1). This function only computes compositions of reduced binary quadratic forms. The fastest algorithm to compute the class group composition is NUCOMP [Sha89]. NUCOMP has a complexity $\mathcal{O}(\mu(n) \cdot n)$ on n-bit discriminants, where $\mu(\cdot)$ denotes the complexity of the applied integer multiplication algorithm. Using an FFT-based multiplication algorithm NUCOMP and thus, this hash function achieves the asymptotic complexity $\mathcal{O}(\log^2|\Delta| \log\log|\Delta|)$. However, we expect concretely better performance from the CSIDH hash than the Wesolowski hash, cf. Figure 8, even though they have the same asymptotic complexity. Multiplying two small norm elements is significantly more efficient than multiplying two random elements. Thus, a natural optimization is to apply a memory-runtime tradeoff, whereby one precomputes all 2^k products of small norm prime forms for a sliding window of length $k \approx 4 - 6$. Afterwards, these precomputed products could be stored in a lookup table. In the online evaluation phase, the precomputed values are read from the lookup table and multiplied together. Alternatively, one might consider not only binary exponents in Step 2 of Figure 4 but longer ones to shorten the necessary number of the generating ideals. We did not explore these optimizations in our implementation, and we leave these promising optimization techniques for future work.

Wesolowski hash (cf. Section 4.2). The two main computationally intensive parts of Wesolowski's original hash function are the hashing to prime parts and the Tonelli-Shanks algorithm for computing modular square roots. The Miller-Rabin primality testing algorithm [Mil75] implemented with an FFT-based multiplication algorithm has a complexity $\mathcal{O}(k \log^2 p \log\log p)$, where k is the number of rounds performed to achieve 2^{-2k} soundness error. The Tonelli-Shanks algorithm has a $\mathcal{O}(\log^2 p)$ expected complexity. Since the randomly sampled prime $p \approx \sqrt{|\Delta|}$, therefore the Wesolowski hash has an overall $\mathcal{O}(k \log^2|\Delta| \log\log|\Delta|)$ asymptotic complexity.

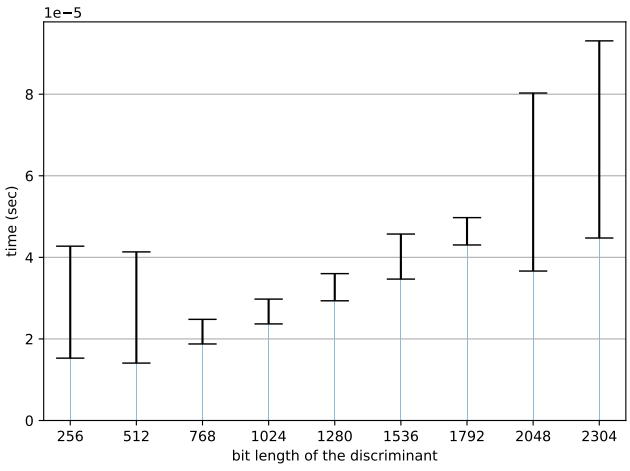
5.2 Empirical Performance Evaluation

We implemented all three proposed hash functions in Python3, and all our source codes are available in our open-source library: `https://www.github.com/seresistvanandras/hashingToClassGroups`. We note that our proof of concept code is highly unoptimized. Therefore, we suggest the reader to consider solely the ratios (when we compare different hash functions) of the following running times and not their absolute values. We ran all our evaluations on a consumer laptop, a MacBook Air (2017), running on a 1,8 GHz Dual-Core Intel Core i5 processor. We note that our implementations are not constant time. We leave it to future work to make our implementations constant time.

Our benchmarks. In our first benchmark, we randomly generated discriminants with bit sizes of 32 bits up to 128 bits to evaluate the hash functions. We hashed $1,000$ random messages for each randomly generated discriminant, cf. Figure 9a. For all discriminants, we applied the Miller-Rabin primality testing [Mil75] in Wesolowski's hash function with 30 iterations corresponding to $\approx 2^{-60}$ soundness error, i.e., that we erroneously generate a composite integer deemed prime. In our second benchmark, we evaluate our hash to class group functions for cryptographically sized discriminants. Specifically, we randomly sampled discriminants of 256-bit length up to 4096-bit length with a step size of 256 bits. For each discriminant, we hashed 100 random messages and reported the average running time of these hash evaluations, cf. Figure 9b. Naturally, in the case of the CSIDH hash function, we did not count

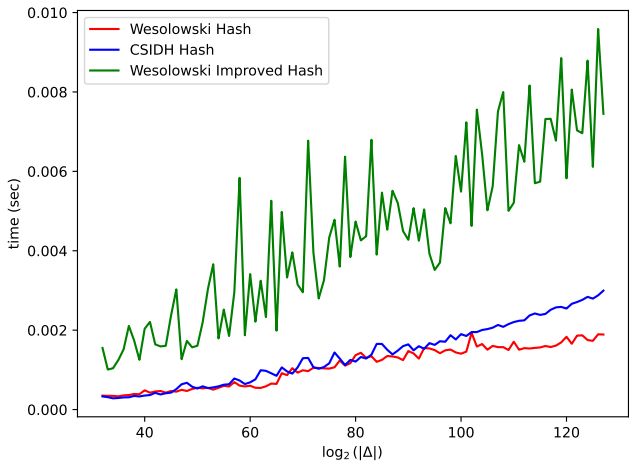


(a) Time to sample a pseudoprime with the Miller-Rabin probabilistic primality testing method ($k = 30$ iteration).

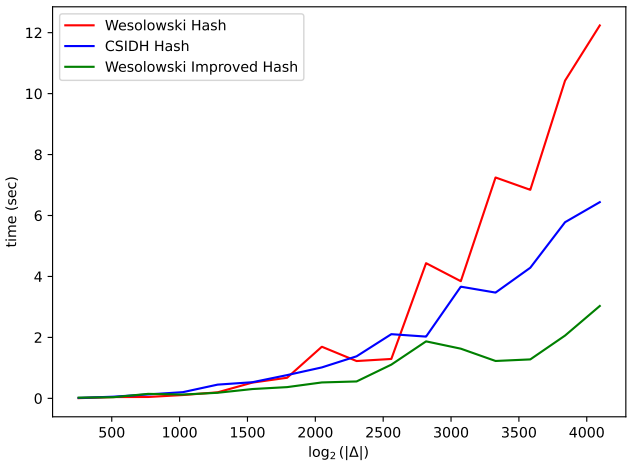


(b) Time to calculate the class group operation in $\mathsf{Cl}(\Delta)$, with the discriminant Δ of given bit size, i.e., from 256 bits to 2304 bits.

Figure 8: We compare the running times of generating a pseudoprime p (soundness error $\approx 2^{-60}$) and calculate the class group operation in $\mathsf{Cl}(\Delta)$ for various cryptographically relevant parameters p and Δ. These two operations are the main bottlenecks of our proposed hash-to-class group functions.



(a) Running time for small discriminants.



(b) Running time for cryptographic parameters.

Figure 9: We evaluate the running time of our proposed hash functions to class groups. We average the running time of 1000 hashing operations for both hash functions from 32 bits up to 128 bits discriminants (left). We average the running time of 100 hashing operations for both hash functions up to 4096 bits discriminants (right).

the one-time preprocessing step's running time. Specifically, in the preprocessing step, we generated $2\log|\Delta|$ small prime-norm ideals satisfying the requirement of Theorem 6.

For small parameters, the Wesolowski hash is faster than the CSIDH hash, cf. Figure 9a. However, we observe in the experimental running times for *cryptographically secure parameters* that Wesolowski's hash-to-class group function is significantly slower than the one inspired by [CLM+18]. We attribute this to the cost of primality testing dominating the running times for larger discriminants. Nevertheless, in cryptographically relevant parameters, both constructions are unbearably slow. In particular, in a class group with 4096-bit discriminant, the Wesolowski hash would take ≈ 10 seconds on average. Interestingly, our improved version of Wesolowski's hash function, cf. Figure 6, outperforms both previous folklore hash functions for all cryptographically secure discriminants (1024 bits and more), see Figure 9b. We leave it to future work as an exciting research direction to improve the efficiency of these proposed hash functions that will be necessary for real-world cryptographic deployments.

6 Conclusion and Future Directions

In this work, we studied the problem of cryptographically securely hashing into class groups of imaginary quadratic fields. We have shown that several widely used open-source libraries apply insecure hashing algorithms that might affect the security of cryptographic protocols that use those libraries. Furthermore, we proposed three secure hashing algorithms, proved their security, and extensively evaluated their performance on real-world parameters.

Despite our over-arching analysis, we leave several directions for future work open. On the practical side, designing more performant and secure hash functions into class groups is an interesting open problem. We also left a constant-time implementation of our proposed hash functions as future work. From a theoretical point of view, it would be interesting to characterize in full generality the discrete probability distributions of the coefficients b or c of reduced forms in $\mathsf{Cl}(\Delta)$. Such a piece of information could help design novel hash functions into class groups of imaginary quadratic fields.

Acknowledgements. We thank Antonio Sanso for introducing us to the VDF deployment design of the Chia network. We thank Michael Zhu for inspiring Section 4.3. We thank Sanne van de Ven and Berry Schoenmakers for pointing out an error in the original formulation of Theorem 5. This research was supported by the Ministry of Culture and Innovation and the National Research, Development, and Innovation Office within the Quantum Information National Laboratory of Hungary (Grant No. 2022-2.1.1-NL-2022-00004). Péter Kutas is supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences and by the UNKP-23-5 New National Excellence Program. Péter Kutas is also partly supported by EPSRC through grant number EP/V011324/1.

References

[ADFMP20] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26*, pages 411–439. Springer, 2020. *[page 11.]*

[AGL+22] Arasu Arun, Chaya Ganesh, Satya Lokam, Tushar Mopuri, and Sriram Sridhar. Dew: transparent constant-sized zksnarks. *Cryptology ePrint Archive*, 2022. *[pages 1 and 2.]*

[AGR+16] Martin Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 191–219. Springer, 2016. *[page 14.]*

[Bac88] Eric Bach. How to generate factored random numbers. *SIAM Journal on Computing*, 17(2):179–193, 1988. *[pages 12, 13, and 14.]*

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 757–788. Springer, Heidelberg, August 2018. *[pages 1 and 2.]*

[BBD+22] Jeremy Booher, Ross Bowden, Javad Doliskani, Tako Boris Fouotsa, Steven D Galbraith, Sabrina Kunzweiler, Simon-Philipp Merz, Christophe Petit, Benjamin Smith, Katherine E Stange, et al. Failing to hash into supersingular isogeny graphs. *arXiv preprint arXiv:2205.00135*, 2022. *[page 2.]*

[BBF19] Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to iops and stateless blockchains. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part I 39*, pages 561–586. Springer, 2019. *[pages 2 and 6.]*

[BCIL23] Cyril Bouvier, Guilhem Castagnos, Laurent Imbert, and Fabien Laguillaumie. I want to ride my bicycl: Bicycl implements cryptography in class groups. *Journal of Cryptology*, 36(3):17, 2023. *[page 2.]*

[BDF+23] Jakob Burkhardt, Ivan Damgård, Tore Kasper Frederiksen, Satrajit Ghosh, and Claudio Orlandi. Improved distributed rsa key generation using the miller-rabin test. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 2501–2515, 2023. *[page 1.]*

[BF97] Dan Boneh and Matthew Franklin. Efficient generation of shared rsa keys. In *Advances in Cryptology—CRYPTO'97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings 17*, pages 425–439. Springer, 1997. *[page 1.]*

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001. *[page 1.]*

[BFS20a] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Heidelberg, May 2020. *[page 1.]*

[BFS20b] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent snarks from dark compilers. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39*, pages 677–706. Springer, 2020. *[page 2.]*

[BJJS10] Jean-François Biasse, Michael J Jacobson Jr, and Alan K Silvester. Security estimates for quadratic field based cryptosystems. In *Australasian Conference on Information Security and Privacy*, pages 233–247. Springer, 2010. *[page 5.]*

[BKSW20] Karim Belabas, Thorsten Kleinjung, Antonio Sanso, and Benjamin Wesolowski. A note on the low order assumption in class group of an imaginary quadratic number fields. *Cryptology ePrint Archive*, 2020. *[page 2.]*

[BKV19] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. Csi-fish: efficient isogeny based signatures through class group computations. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 227–247. Springer, 2019. *[pages 5 and 12.]*

[BKW20] Dan Boneh, Dmitry Kogan, and Katharine Woo. Oblivious pseudorandom functions from isogenies. In *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26*, pages 520–550. Springer, 2020. *[page 11.]*

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *International conference on the theory and application of cryptology and information security*, pages 514–532. Springer, 2001. *[page 1.]*

[BN00] Dan Boneh and Moni Naor. Timed commitments. In *Annual international cryptology conference*, pages 236–254. Springer, 2000. *[pages 1 and 2.]*

[BV07] Johannes Buchmann and Ulrich Vollmer. *Binary quadratic forms*. Springer, 2007. *[pages 3, 4, and 5.]*

[BW88] Johannes Buchmann and Hugh C. Williams. A key-exchange system based on imaginary quadratic fields. *Journal of Cryptology*, 1:107–118, 1988. *[pages 1 and 4.]*

[CHI+21] Megan Chen, Carmit Hazay, Yuval Ishai, Yuriy Kashnikov, Daniele Micciancio, Tarik Riviere, Abhi Shelat, Muthu Venkitasubramaniam, and Ruihan Wang. Diogenes: Lightweight scalable rsa modulus generation with a dishonest majority. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 590–607. IEEE, 2021. *[page 1.]*

[CL15] Guilhem Castagnos and Fabien Laguillaumie. Linearly homomorphic encryption from. In *Cryptographers' Track at the RSA Conference*, pages 487–505. Springer, 2015. *[page 1.]*

[CLM+18] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. Csidh: an efficient post-quantum commutative group action. In *Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part III 24*, pages 395–427. Springer, 2018. *[pages 3, 11, and 16.]*

[CMB23] Kevin Choi, Aathira Manoj, and Joseph Bonneau. Sok: Distributed randomness beacons. *Cryptology ePrint Archive*, 2023. *[page 2.]*

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In *Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18*, pages 402–414. Springer, 1999. *[page 3.]*

[CP19] Bram Cohen and Krzysztof Pietrzak. The chia network blockchain. *White Paper, Chia. net*, 9, 2019. *[page 2.]*

[DFI95] William Duke, John B Friedlander, and Henryk Iwaniec. Equidistribution of roots of a quadratic congruence to prime moduli. *Annals of Mathematics*, 141(2):423–441, 1995. *[page 22.]*

[DFMPS19] Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. Verifiable delay functions from supersingular isogenies and pairings. In *Advances in Cryptology–ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part I 25*, pages 248–277. Springer, 2019. *[page 2.]*

[DGS20] Samuel Dobson, Steven D Galbraith, and Benjamin Smith. Trustless groups of unknown order with hyperelliptic curves. *IACR Cryptol. ePrint Arch.*, 2020:196, 2020. *[pages 4, 5, and 10.]*

[Dix08] John D Dixon. Generating random elements in finite groups. *the electronic journal of combinatorics*, pages R94–R94, 2008. *[page 11.]*

[FFS+13] Reza R Farashahi, Pierre-Alain Fouque, Igor Shparlinski, Mehdi Tibouchi, and J Voloch. Indifferentiable deterministic hashing to elliptic and hyperelliptic curves. *Mathematics of Computation*, 82(281):491–512, 2013. *[page 1.]*

[FHSS+23] Armando Faz-Hernandez, Sam Scott, Nick Sullivan, Riad S. Wahby, and Christopher A. Wood. Hashing to Elliptic Curves. RFC 9380, August 2023. *[page 1.]*

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986. *[page 14.]*

[FSV09] Reza R Farashahi, Igor E Shparlinski, and José Felipe Voloch. On hashing into elliptic curves. *Journal of Mathematical Cryptology*, 3(4):353–360, 2009. *[page 1.]*

[FT12] Pierre-Alain Fouque and Mehdi Tibouchi. Indifferentiable hashing to barreto–naehrig curves. In *Progress in Cryptology–LATINCRYPT 2012: 2nd International Conference on Cryptology and Information Security in Latin America, Santiago, Chile, October 7-10, 2012. Proceedings 2*, pages 1–17. Springer, 2012. *[page 1.]*

[GKL+22] Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch. Reinforced concrete: A fast hash function for verifiable computation. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1323–1335, 2022. *[page 14.]*

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for {Zero-Knowledge} proof systems. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 519–535, 2021. *[page 14.]*

[GRR+16] Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P Smart. Mpc-friendly symmetric key primitives. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 430–443, 2016. *[page 14.]*

[GS98] David M Goldschlag and Stuart G Stubblebine. Publicly verifiable lotteries: Applications of delaying functions. In *International Conference on Financial Cryptography*, pages 214–226. Springer, 1998. *[page 2.]*

[HM89] James L Hafner and Kevin S McCurley. A rigorous subexponential algorithm for computation of class groups. *Journal of the American mathematical society*, 2(4):837–850, 1989. *[page 5.]*

[HMR+19] Carmit Hazay, Gert Læssøe Mikkelsen, Tal Rabin, Tomas Toft, and Angelo Agatino Nicolosi. Efficient rsa key generation and threshold paillier in the two-party setting. *Journal of Cryptology*, 32:265–323, 2019. *[page 1.]*

[Ica09] Thomas Icart. How to hash into elliptic curves. In *Annual International Cryptology Conference*, pages 303–316. Springer, 2009. *[pages 1 and 3.]*

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography: principles and protocols.* Chapman and hall/CRC, 2007. *[page 1.]*

[KPS18] Ahmed Kosba, Charalampos Papamanthou, and Elaine Shi. xjsnark: A framework for efficient verifiable computation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 944–961. IEEE, 2018. *[page 14.]*

[Lip12] Helger Lipmaa. Secure accumulators from euclidean rings without trusted setup. In *Applied Cryptography and Network Security: 10th International Conference, ACNS 2012, Singapore, June 26-29, 2012. Proceedings 10*, pages 224–240. Springer, 2012. *[page 1.]*

[Lon19] Jieyi Long. Nakamoto consensus with verifiable delay puzzle. *arXiv preprint arXiv:1908.06394*, 2019. *[page 2.]*

[LSS20] Esteban Landerreche, Marc Stevens, and Christian Schaffner. Non-interactive cryptographic timestamping based on verifiable delay functions. In *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24*, pages 541–558. Springer, 2020. *[page 2.]*

[Mil75] Gary L Miller. Riemann's hypothesis and tests for primality. In *Proceedings of the seventh annual ACM symposium on Theory of computing*, pages 234–239, 1975. *[page 15.]*

[MOT20] Tomoki Moriya, Hiroshi Onuki, and Tsuyoshi Takagi. Sigamal: a supersingular isogeny-based pke and its application to a prf. In *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26*, pages 551–580. Springer, 2020. *[page 11.]*

[MT19] Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. Homomorphic time-lock puzzles and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 620–649. Springer, Heidelberg, August 2019. *[page 1.]*

[Pea92] Karl Pearson. *On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling.* Springer, 1992. *[page 6.]*

[Pie19] Krzysztof Pietrzak. Simple verifiable delay functions. In *10th innovations in theoretical computer science conference (itcs 2019)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2019. *[pages 1 and 2.]*

[RSS20] Lior Rotem, Gil Segev, and Ido Shahaf. Generic-group delay functions require hidden-order groups. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 155–180. Springer, 2020. *[page 2.]*

[RSW96] Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. 1996. *[page 1.]*

[SB23] István András Seres and Péter Burcsi. Behemoth: transparent polynomial commitment scheme with constant opening proof size and verifier time. *Cryptology ePrint Archive*, 2023. *[pages 1 and 2.]*

[SBC+09] Michael Scott, Naomi Benger, Manuel Charlemagne, Luis J Dominguez Perez, and Ezekiel J Kachisa. Fast hashing to g 2 on pairing-friendly curves. In *International Conference on Pairing-Based Cryptography*, pages 102–113. Springer, 2009. *[page 1.]*

[Sha89] Daniel Shanks. On gauss and composition i, ii. In *Proc. NATO ASI on Number Theory and Applications*, pages 163–179. Kluwer Academic Press Dordrecht, 1989. *[page 15.]*

[SL84] C-P Schnorr and Hendrik W Lenstra. A monte carlo factoring algorithm with linear storage. *Mathematics of Computation*, 43(167):289–311, 1984. *[page 2.]*

[Sou07] K Soundararajan. The number of imaginary quadratic fields with a given class number. *Hardy-Ramanujan Journal*, 30, 2007. *[page 5.]*

[Sut07] Andrew V Sutherland. *Order computations in generic groups*. PhD thesis, Massachusetts Institute of Technology, 2007. *[page 5.]*

[TCLM21] Sri Aravinda Krishnan Thyagarajan, Guilhem Castagnos, Fabian Laguillaumie, and Giulio Malavolta. Efficient cca timed commitments in class groups. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2663–2684, 2021. *[pages 1, 2, and 6.]*

[WB19] Riad S Wahby and Dan Boneh. Fast and simple constant-time hashing to the bls12-381 elliptic curve. *Cryptology ePrint Archive*, 2019. *[page 1.]*

[Wes19] Benjamin Wesolowski. Efficient verifiable delay functions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 379–407. Springer, Heidelberg, May 2019. *[pages 1, 2, 3, 11, and 12.]*

A The coefficient c of reduced forms

In the following, we aim to characterize heuristically the asymptotic behavior of the discrete probability distribution of the reduced binary quadratic forms' c coefficient in $\mathsf{Cl}(\Delta)$. We introduce a similar definition to Definition 9.

Definition 10. *(The distribution of coefficient c in $\mathsf{Cl}(\Delta)$) Let us denote by $\mathcal{C}_\Delta$ the discrete probability distribution of the coefficient c of the reduced forms in $\mathsf{Cl}(\Delta)$, i.e.,*

$$\Pr[\mathcal{C}_\Delta = c] := \frac{|\{(a,b)|\exists (a,b) : (a,b,c) \in \mathsf{Cl}(\Delta)\}|}{|\mathsf{Cl}(\Delta)|}. \tag{17}$$

Let $C_{(a,b)} := \{(a,b)|(a,b,c) \in \mathsf{Cl}(\Delta)\}$, i.e., recall this condition implies that $c \mid \frac{b^2-\Delta}{4} \wedge |b| \leq a \leq \sqrt{\frac{|\Delta|}{3}} \wedge a \leq c$. We want to obtain close estimates on $|C_a|$ for a fixed, large Δ.

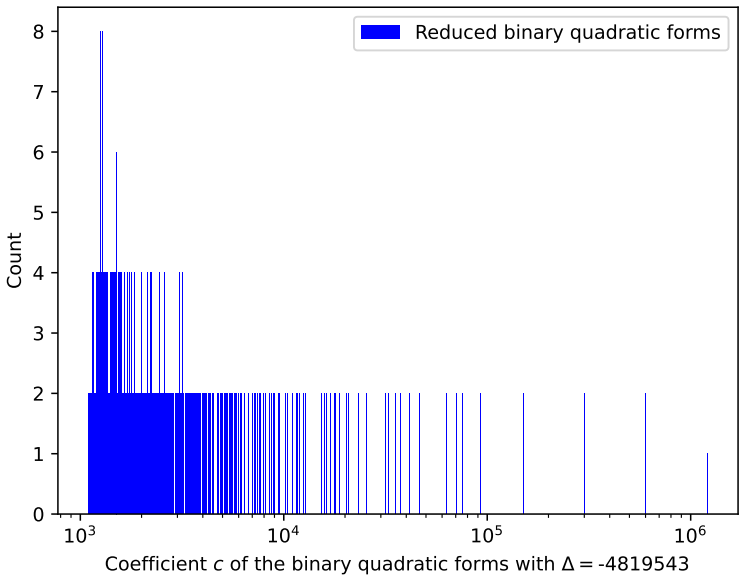


Figure 10: Illustrating the distribution of the coefficient a and b of reduced binary quadratic forms in the class group $\mathsf{Cl}(\Delta)$ with the medium-sized discriminant $\Delta = -4,819,543$.

For the sake of simplicity, let us only consider prime forms in C_a, i.e., $a = p$ for a prime p. Recall Theorem 1, that is $C_p \neq \emptyset$ iff. $\left(\frac{\Delta}{p}\right) = 1$. If $C_p \neq \emptyset$, then one can compute the b coefficients of the prime form (p, b) as the two modular square roots, i.e., $b \leftarrow \mathsf{SqrtModP}(\Delta, p)$, cf. Theorem 2. Applying the equidistribution theorems of Duke, Friedlander, and Iwaniec [DFI95], we assert that the rationals $\mathcal{X} \sim \frac{b}{p}$ is equidistributed in $[0, 1]$ for $p \in [0, \sqrt{|\Delta|/4}]$ and $b < p$. Therefore, we have the distribution $\mathcal{X}^2 \sim \frac{b^2}{p^2}$ has $F_{\mathcal{X}^2}(x) = \sqrt{x} \in [0, 1]$, and the density function of $\mathcal{X}^2$ is $f_{\mathcal{X}^2}(x) = \frac{dF_{\mathcal{X}^2}(x)}{dx} = \frac{1}{2\sqrt{x}}$.

Looking ahead, we wish to establish heuristically the distribution of $c = \frac{b^2-\Delta}{4p}$. Our goal is to obtain a heuristic characterization of the cumulative distribution function for the coefficient c of reduced prime forms. More formally,

$$F_{\mathcal{C}_\Delta}(x) := \Pr[\mathcal{C}_\Delta \leq x] = \Pr\left[\frac{b^2 - \Delta}{4p} \leq x : \left(\frac{\Delta}{p}\right) = 1 \wedge p \in [2, \sqrt{|\Delta|/4}] \wedge b/p \sim U(0, 1)]\right]. \tag{18}$$

Let $\mathcal{P} := \{p | p \text{ prime} \wedge \left(\frac{\Delta}{p}\right) = 1 \wedge p \in [2, \sqrt{|\Delta|/4}]\}$. In Equation (18), we can heuristically think of the probability $\Pr[\frac{b^2-\Delta}{4p} \leq x]$ as follows: pick $p \in_R \mathcal{P}$ and we pretend that b/p is a uniform random variable in $[0, 1]$ (even though p determines b up to sign). Rearranging terms, we obtain:

$$\Pr\left[\frac{b^2 - \Delta}{4p} \leq x\right] = \Pr\left[\frac{b^2 - \Delta}{4p^2} p \leq x\right] = \Pr\left[\frac{p}{4}\left(\frac{b}{p}\right)^2 - \frac{\Delta}{4p} \leq x\right] \tag{19}$$

Expressing b/p in terms of the other terms, the expression becomes:

$$\Pr\left[\left(\frac{b}{p}\right)^2 \leq \frac{4x}{p} + \frac{\Delta}{p^2}\right] = \Pr\left[\mathcal{X}^2 \leq \frac{4x}{p} + \frac{\Delta}{p^2}\right] =: \mathcal{G}_{x,p}. \tag{20}$$

We observe the following elementary bounds for coefficient c. Recall $b \leq a \leq c$. Thus, $4c^2 \geq 4ac = b^2 - \Delta \geq -\Delta$. Therefore, $4c^2 \geq -\Delta$, i.e., $c \geq \frac{\sqrt{|\Delta|}}{2}$. Therefore, we focus on estimating the distribution function for $x \geq |\Delta|^{1/2+\varepsilon}$ for $\varepsilon > 0$. For such an x, we have $p > |\Delta|^{1/2-\varepsilon}$ if $0 \leq \frac{4x}{p} + \frac{\Delta}{p^2}$, cf. Equation (20). The probability $\mathcal{G}_{x,p} = 0$ for $p < \frac{-\Delta}{4x}$, and $\mathcal{G}_{x,p} = 1$ for $p > 2x - \sqrt{4x^2 + \Delta} \approx \frac{-\Delta}{4x}$ for x in this range. Thus, the expected value of $\mathcal{G}_{x,p} = 1$ for a random p is approximately equal to the probability that $p > \frac{-\Delta}{4x}$. Since p is a random prime from the interval $[2, \sqrt{\frac{|\Delta|}{4}}]$, we obtain

$$F_{\mathcal{C}_\Delta}(x) \approx 1 - \frac{\frac{|\Delta|}{4x}/\log\frac{|\Delta|}{4x}}{\sqrt{\frac{|\Delta|}{4}}/\log\sqrt{\frac{|\Delta|}{4}}} = 1 - \Theta\left(\frac{\sqrt{|\Delta|}}{x}\right) \tag{21}$$

The last step is due to the fact that the quotient of the logarithmic terms only contributed a small constant factor. Backed by the previous heuristics for prime forms and based on empirical investigation, we formulate the following conjecture for *all* binary quadratic forms' coefficient c in $\mathsf{Cl}(\Delta)$.

Conjecture 2. *The cumulative distribution function $G_{\mathcal{C}_\Delta}(\cdot)$ of* all *reduced binary quadratic forms' coefficient c in $\mathsf{Cl}(\Delta)$, i.e., the distribution $\mathcal{C}_\Delta$ for $x \geq |\Delta|^{1/2+\varepsilon}$ follows*

$$G_{\mathcal{C}_\Delta}(x) = 1 - \Theta\left(\frac{\sqrt{|\Delta|}}{x}\right). \tag{22}$$