Cryptographic Smooth Neighbors

Giacomo Bruno[1], Maria Corte-Real Santos[2⋆], Craig Costello[3], Jonathan Komada Eriksen[4], Michael Meyer[5⋆⋆], Michael Naehrig[3], and Bruno Sterner[6⋆⋆⋆]

[1] IKARUS Security Software
giako13@gmail.com
[2] University College London
maria.santos.20@ucl.ac.uk
[3] Microsoft Research, USA
{craigco,mnaehrig}@microsoft.com
[4] Norwegian University of Science and Technology
jonathan.k.eriksen@ntnu.no
[5] University of Regensburg, Germany
michael@random-oracles.org
[6] Surrey Centre for Cyber Security, University of Surrey, UK
b.sterner@surrey.ac.uk

Abstract. We revisit the problem of finding two consecutive B-smooth integers by giving an optimised implementation of the Conrey-Holmstrom-McLaughlin "smooth neighbors" algorithm. While this algorithm is not guaranteed to return the complete set of B-smooth neighbors, in practice it returns a very close approximation to the complete set, but does so in a tiny fraction of the time of its exhaustive counterparts. We exploit this algorithm to find record-sized solutions to the pure twin smooth problem. Though these solutions are still not large enough to be cryptographic parameters themselves, we feed them as input into known methods of searching for twins to yield cryptographic parameters that are much smoother than those given in prior works. Our methods seem especially well-suited to finding parameters for the SQISign signature scheme, particularly those that are geared towards high-security levels.

Keywords: Post-quantum cryptography, isogeny-based cryptography, twin smooth integers, smooth neighbors, Pell equation, SQISign.

1 Introduction

In recent years the tantalising problem of finding two large, consecutive, smooth integers has emerged in the context of instantiating efficient isogeny-based public key cryptosystems. Though the problem was initially motivated in the context of key exchange [8], a wave of polynomial time attacks [5,18,19] has completely broken the isogeny-based key exchange scheme SIDH [14], leaving post-quantum signatures as the most compelling cryptographic application of isogenies at present. In terms of practical potential, the leading isogeny-based signature scheme is SQISign [12]; it boasts the smallest public keys and signatures of all post-quantum signature schemes (by far!), at the price of a signing algorithm that is orders of magnitude slower than its post-quantum counterparts. Finding secure parameters for SQISign is related to the twin smooth problem mentioned above, with a large contributing factor to the overall efficiency of the protocol being the smoothness bound, B, of the rational torsion used in isogeny computations. This bound corresponds to the degree of the largest prime-degree isogeny computed in the protocol, for which the fastest algorithm runs in $\tilde{O}(\sqrt{B})$ field operations [2]. Part of the reason for SQISign's performance drawback is that the problem of finding parameters with small B is difficult: the fastest implementation to date targets security comparable to NIST Level I [23, §4.A] and has $B = 3923$ [13].

[⋆] Supported by EPSRC grant EP/S022503/1.

[⋆⋆] Supported by the German Federal Ministry of Education and Research (BMBF) under the project QuantumRISC (ID 16KIS1039).

[⋆⋆⋆] Supported by EPSRC grant EP/R513350/1.

The CHM algorithm. In this work we introduce new ways of finding large twin smooth instances based on the Conrey-Holmstrom-McLaughlin (CHM) "Smooth neighbors" algorithm [7]. For a fixed smoothness bound B, the CHM algorithm starts with the set of integers $S = \{1, 2, \ldots, B-1\}$ representing the smooth neighbors $(1, 2), (2, 3), \ldots, (B-1, B)$, and recursively grows this set by constructing new twin smooth integers from unordered pairs in $S \times S$ until a full pass over all such pairs finds no new twins, at which point the algorithm terminates. Although the CHM algorithm is not guaranteed to find the set of all B-smooth twins, for moderate values of B it converges with the set S containing *almost all* such twins. The crucial advantage is that, unlike the algorithm of Lehmer [16] that exhaustively solves $2^{\pi(B)}$ Pell equations to guarantee the full set of B-smooth twins, the CHM algorithm terminates much more rapidly. For example, in 2011 Luca and Najman [17] used Lehmer's approach with $B = 100$ to compute the full set of 13,374 twin smooths in 15 days (on a quad-core 2.66 GHz processor) by solving $2^{\pi(B)} = 2^{25}$ Pell equations, the solutions of which can have as many as 10^{10^6} decimal digits. The largest pair of 100-smooth twins they found were the 58-bit integers

$$\begin{aligned}
166055401586083680 &= 2^5 \cdot 3^3 \cdot 5 \cdot 11^3 \cdot 23 \cdot 43 \cdot 59 \cdot 67 \cdot 83 \cdot 89, \text{ and} \\
166055401586083681 &= 7^2 \cdot 17^{10} \cdot 41^2.
\end{aligned}$$

In 2012, Conrey, Holmstrom and McLaughlin ran *their* algorithm on a similar machine to find 13,333 (i.e. all but 41) of these twins in 20 minutes [7]. Subsequently, they set $B = 200$ and found a list of 346,192 twin smooths in about 2 weeks, the largest of which were the 79-bit integers

$$\begin{aligned}
589864439608716991201560 &= 2^3 \cdot 3^3 \cdot 5 \cdot 7^2 \cdot 11^2 \cdot 17 \cdot 31 \cdot 59^2 \cdot 83 \cdot 139^2 \\
&\quad \cdot 173 \cdot 181, \text{ and} \\
589864439608716991201561 &= 13^2 \cdot 113^2 \cdot 127^2 \cdot 137^2 \cdot 151^2 \cdot 199^2.
\end{aligned}$$

Exhausting the full set of 200-smooth twins would have required solving $2^{\pi(200)} = 2^{46}$ Pell equations, which is pushing the limit of what is currently computationally feasible. The largest run of Lehmer's algorithm reported in the literature used $B = 113$ [8, §5.3], which required solving 2^{30} Pell equations and a significant parallelized computation that ran over several weeks. The largest set of 113-smooth twins found during that computation were the 75-bit integers

$$\begin{aligned}
19316158377073923834000 &= 2^4 \cdot 3^6 \cdot 5^3 \cdot 7 \cdot 23^2 \cdot 29 \cdot 47 \cdot 59 \cdot 61 \cdot 73 \cdot 97 \cdot 103, \\
19316158377073923834001 &= 13^2 \cdot 31^2 \cdot 37^2 \cdot 43^4 \cdot 71^4.
\end{aligned}$$

Remark 1. The above examples illustrate some important phenomena that are worth pointing out before we move forward. Observe that, in the first and third examples, the largest prime not exceeding B is not found in the factors of the largest twins. The largest 89-smooth twins are the same as the largest 97-smooth twins, and the largest 103-smooth twins are the same as the largest 113-smooth twins. In other words, increasing B to include more primes necessarily increases the size of the set of B-smooth twins, but it does not mean we will find any new, larger twins. This trend highlights part of the difficulty we face in trying to find optimally smooth parameters of cryptographic size: increasing the smoothness bound B makes the size of the set of twins grow rapidly, but the growth of the largest twins we find is typically painstakingly slow. The set of 100-smooth twins has cardinality 13,374, with the largest pair being 58 bits; increasing B to 200 gives a set of cardinality (at least) 345,192, but the largest pair has only grown to be 79 bits. In fact, most of this jump in the bitlength of the largest twins occurs when increasing $B = 97$ (58 bits) to include two more primes with $B = 103$ (76 bits). Including the 19 additional primes up to 199 only increases the bitlength of largest twins with $B = 199$ by 3 (79 bits), and this is indicative of what we observe when B is increased even further.

Our contributions. We give an optimised implementation of CHM that allows us to run the algorithm for much larger values of B in order to find larger sized twins. For example, with

$B = 547$, our implementation converged with a set of 82,026,426 pairs of B-smooth twins, the largest of which are the 122-bit pair $(r, r+1)$ with

$$\begin{aligned} r &= 5^4 \cdot 7 \cdot 13^2 \cdot 17^2 \cdot 19 \cdot 29 \cdot 41 \cdot 109 \cdot 163 \cdot 173 \cdot 239 \cdot 241^2 \cdot 271 \cdot 283 \\ &\quad \cdot 499 \cdot 509, \qquad \text{and} \\ r+1 &= 2^8 \cdot 3^2 \cdot 31^2 \cdot 43^2 \cdot 47^2 \cdot 83^2 \cdot 103^2 \cdot 311^2 \cdot 479^2 \cdot 523^2. \end{aligned} \tag{1}$$

Although it remains infeasible to increase B to the point where the twins found through CHM are large enough to be used out-of-the-box in isogeny-based schemes (i.e. close to 2^{256}), we are able to combine the larger twins found through CHM with techniques from the literature in order to find much smoother sets of SQISign parameters. In this case we are aided by the requirements for SQISign, which permit us to relax the size of the smooth factor that divides p^2-1. The current state-of-the-art instantiation [13] uses primes p such that

$$\ell^f \cdot T \mid (p^2-1),$$

where ℓ is a small prime (typically $\ell = 2$), where f is as large as possible, and where $T \approx p^{5/4}$ is both coprime to ℓ and B-smooth. For example, the original SQISign implementation [12] used a 256-bit prime p such that

$$p^2 - 1 = 2^{34} \cdot T_{1879} \cdot R,$$

where T_{1879} is an odd 334-bit integer[7] whose largest prime factor is $B = 1879$, and R is the *rough* factor; a 144-bit integer containing no prime factors less than or equal to B. As another example, De Feo, Leroux and Wesolowski [13, §5] instead use a 254-bit prime p with

$$p^2 - 1 = 2^{66} \cdot T_{3923} \cdot R,$$

where T_{3923} is an odd 334-bit integer whose largest prime factor is $B = 3923$, and where all of R's prime factors again exceed B.

During the search mentioned above that found the record 547-smooth twins in (1), over 82 million other pairs of smaller sized twins were found. One such pair was the 85-bit twins $(r, r+1)$ with $r = 26606682403634464748953600$. Taking $p = 2r^3 - 1$ gives a 255-bit prime p such that

$$p^2 - 1 = 2^{41} \cdot T_{547} \cdot R,$$

where T is an odd 327-bit integer that is 547-smooth. This represents a significant improvement in smoothness over the T values obtained in [12] and [13]. Although the smoothness of T is not the only factor governing the efficiency of the scheme, our analysis in Section 6 suggests that the parameters found in this paper compare favorably with those currently found in SQISign implementations.

Just as we transformed a pair of 85-bit twins into a 255-bit prime by taking $p = 2r^3 - 1$, we combine the use of twins found with CHM and primes of the form $p = 2r^n - 1$ with $n \geq 3$ to obtain several SQISign-friendly primes that target higher security levels. For example, with some 64-bit twins $(r, r+1)$ found through CHM, we give a 382-bit prime $p = 2r^6 - 1$ such that $p^2 - 1 = 2^{80} \cdot T_{10243} \cdot R$, where T is an odd 495-bit integer that is 10243-smooth; this prime would be suitable for SQISign signatures geared towards NIST Level III security. As another example, with some 85-bit twins $(r, r+1)$, we give a 508-bit prime $p = 2r^6 - 1$ such that $p^2 - 1 = 2^{86} \cdot T_{150151} \cdot R$, where T is a 639-bit integer that is 150151-smooth; this prime would be suitable for SQISign signatures targeting NIST Level V security.

Our implementation of the CHM algorithm is written in C/C++ and is found at

https://github.com/GiacomoBruno/TwinsmoothSearcher.

[7] The initial SQISign requirements [12] had $T \approx p^{3/2}$, but T_{1879} corresponds to the new requirements.

Organisation. In Section 2 we review prior methods for generating large instances of twin smooths and recall some results on smoothness probabilities. In Section 3 we recall the CHM algorithm and give a generalisation of it that may be of independent interest. In Section 4 we detail our implementation of the CHM algorithm and present a number of optimisations that allowed us to run it for much larger values of B. In Section 5 we discuss the combination of CHM with primes of the form $p = 2x^n - 1$ to give estimates on the probabilities of finding SQISign parameters at various security levels. In Section 6 we present our results, giving record-sized twin smooth instances as well as dozens of SQISign-friendly primes that target NIST's security levels I, III, and V.

2 Preliminaries

In §2.1 we review prior methods of searching for twin smooth integers, and in §2.2 we recall some theory concerning smoothness probabilities.

We start by fixing some definitions and terminology.

Definition 1. *A positive integer n is called* B-smooth *for some real number $B > 0$ if all prime divisors of n are at most B. An integer n* generates a B-smooth value *of a polynomial $f(X)$ if $f(n)$ is B-smooth. In this case we call n a B-smooth value of $f(X)$. We call two consecutive integers* B-smooth twins *if their product is B-smooth. An integer n is called* B-rough *if all of its prime factors exceed B.*

2.1 Prior Methods of Searching for Large Twin Smooth Integers

We follow the descriptions of the three algorithms reviewed in [9, §2] as well as including the method introduced in [9] itself.

Solving Pell equations. Fix B, let $\{2, 3, \ldots, q\}$ be the set of primes up to B with cardinality $\pi(B)$, and consider the B-smooth twins $(r, r+1)$. Let $x = 2r + 1$, so that $x - 1$ and $x + 1$ are also B-smooth, and let D be the squarefree part of their product $(x-1)(x+1)$, i.e. $x^2 - 1 = Dy^2$ for some $y \in \mathbb{Z}$. It follows that Dy^2 is B-smooth, which means that

$$D = 2^{\alpha_2} \cdot 3^{\alpha_3} \cdot \cdots \cdot q^{\alpha_q}$$

with $\alpha_i \in \{0, 1\}$ for $i = 2, 3, \ldots, q$. For each of the $2^{\pi(B)}$ squarefree possibilities for D, Størmer [20] reverses the above argument and proposes to solve the $2^{\pi(B)}$ Pell equations

$$x^2 - Dy^2 = 1,$$

finding *all* of the solutions for which y is B-smooth, and in doing so finding the complete set of B-smooth twins.

The largest pair of 2-smooth integers is $(1, 2)$, the largest pair of 3-smooth integers is $(8, 9)$, and the largest pair of 5-smooth integers is $(80, 81)$. Unfortunately, solving $2^{\pi(B)}$ Pell equations becomes infeasible before the size of the twins we find is large enough (i.e. exceeds 2^{200}) for our purposes. As we saw in Section 1, [8] reports that with $B = 113$ the largest twins $(r, r+1)$ found upon solving all 2^{30} Pell equations have $r = 19316158377073923834000 \approx 2^{75}$.

The extended Euclidean algorithm. The most naïve way of searching for twin smooth integers is to compute B-smooth numbers r until either $r - 1$ or $r + 1$ also turns out to be B-smooth. A much better method [8,12] is to instead choose two coprime B-smooth numbers α and β that are both of size roughly the square root of the target size of r and $r+1$. On input of α and β, Euclid's extended GCD algorithm outputs two integers (s, t) such that $\alpha s + \beta t = 1$ with $|s| < |\beta/2|$ and $|t| < |\alpha/2|$. We can then take $\{m, m+1\} = \{|\alpha s|, |\beta t|\}$, and the probability of m and $m+1$ being B-smooth is now the probability that $s \cdot t$ is B-smooth. The reason this performs much better than the naïve method above is that $s \cdot t$ with $s \approx t$ is much more likely to be B-smooth than a random integer of similar size.

Searching with $r = x^n - 1$. A number of works [8,12,13] have found performant parameters by searching for twins of the form $(r, r+1) = (x^n - 1, x^n)$, for relatively small $n \in \mathbb{Z}$. For example, suppose we are searching for b-bit twins $(r, r+1)$ and we take $n = 4$ so that $r = (x^2+1)(x-1)(x+1)$. Instead of searching for two b-bit numbers that are smooth, we are now searching for three smooth $(b/4)$-bit numbers (i.e. $x - 1$, x, and $x + 1$) and one smooth $(b/2)$-bit number, which increases the probability of success (see [9]).

Searching with PTE solutions. The approach taken in [9] can be viewed as an extension of the method above, where the important difference is that for $n > 2$ the polynomial $x^n - 1$ does not split in $\mathbb{Z}[x]$, and the presence of higher degree terms (like the irreducible quadratic $x^2 + 1$ above) significantly hampers the probability that values of $x^n - 1 \in \mathbb{Z}$ are smooth. Instead, the algorithm in [9] takes $(r, r + 1) = (f(x), g(x))$, where $f(x)$ and $g(x)$ are both of degree n and are comprised entirely of linear factors. This boosts the success probability again, but one of the difficulties facing this method is that polynomials $f(x)$ and $g(x)$ that differ by a constant and are completely split are difficult to construct for $n \geq 4$. Fortunately, instances of these polynomials existed in the literature prior to [9], since they can be trivially constructed using solutions to the Prouhet-Tarry-Escott (PTE) problem (see [9]).

2.2 Smooth (and Rough) Probabilities

In Section 5, we will compute the probabilities of finding integers with large smooth divisor. We follow Banks and Shparlinski [1] to introduce the necessary background for these computations.

We introduce the function $\Theta(X, B, D)$ that counts positive integers $N \leq X$ for which there exists a divisor $d \mid N$ with $d > D$ and such that d is B-smooth. In other words,

$$\Theta(X, B, D) = \#\{N \leq X : D < \text{largest } B\text{-smooth divisor of } N\}.$$

This function has been previously studied in the literature, for example [22,21].

For X, B, D varying over a wide domain, Banks and Shparlinski derive the first two terms of the asymptotic expansion of $\Theta(X, B, D)$. The expansion uses the Dickman function ρ [11,10] and the Buchstab function ω [4], which are related to the counting functions

$$\Psi(X, B) = \#\{N \leq X : N \text{ is } B\text{-smooth}\}, \tag{2}$$
$$\Phi(X, B) = \#\{N \leq X : N \text{ is } B\text{-rough}\}. \tag{3}$$

Indeed, for a large range of X and B, it is known that

$$\Psi(X, B) \sim \rho(u)X \quad \text{and} \quad \Phi(X, B) \sim \omega(u)\frac{X}{\log B},$$

where $u = (\log X)/(\log B)$.

The asymptotic expansion, due to Banks and Shparlinski, is given by the following theorem [1, Theorem 1].

Theorem 1. *For fixed $\epsilon > 0$ and uniformly in the domain*

$$X \geq 3,\ B \geq \exp\{(\log\log X)^{5/2+\epsilon}\},\ B\log B \leq D \leq X/B,$$

we have

$$\Theta(X, B, D) = \Big(\rho(u) + \mathcal{C}_{\omega,\rho}(u, v)\Big)X - \gamma\mathcal{C}_{\omega,\rho'}(u, v)\frac{X}{\log B} + O(\mathcal{E}(X, B, D)),$$

where $u = (\log X)/(\log B)$, $v = (\log D)/(\log B)$, γ is the Euler–Mascheroni constant, $\mathcal{C}_{f,g}$ is the partial convolution of f with g (as defined in [1, pg. 2]), and

$$\mathcal{E}(X, B, D) = \frac{X}{\log B}\left\{\rho(u-1) + \frac{\rho(v)\log(v+1)}{\log B} + \frac{\rho(v)}{\log(v+1)}\right\}.$$

We use this asymptotic expansion to estimate the value of Θ for X, B, D in the correct range.

3 The CHM Algorithm

In this section, we first recall the Conrey, Holmstrom, and McLaughlin (CHM) algorithm [7], a remarkably simple algorithm that generates twin smooth integers (or *smooth neighbors* as they are called in [7]), i.e. smooth values of the polynomial $X(X+1)$. We then present a generalisation of this algorithm, which generates smooth values of any monic quadratic polynomial. The algorithm generalises the CHM algorithm, as well as another algorithm in the literature by Conrey and Holmstrom [6], which generates smooth values of the polynomoial X^2+1. In the end, we are primarily interested in the CHM algorithm, but present the generalised algorithm here, as it may be of independent interest.

3.1 Finding Smooth Twins with the CHM Algorithm

Conrey, Holmstrom, and McLaughlin [7] present the following algorithm for producing many B-smooth values of $X(X+1)$. It starts with the initial set

$$S^{(0)} = \{1, 2, \ldots, B-1\}$$

of all integers less than B, representing the B-smooth twins $(1,2)$, $(2,3)$, …, $(B-1, B)$. Next, it iteratively passes through all pairs of distinct $r, s \in S^{(0)}, r < s$ and computes

$$\frac{t}{t'} = \frac{r}{r+1} \cdot \frac{s+1}{s},$$

writing $\frac{t}{t'}$ in lowest terms. If $t' = t+1$, then clearly t also represents a twin smooth pair. The next set $S^{(1)}$ is formed as the union of $S^{(0)}$ and the set of all solutions t such that $t' = t+1$. Now the algorithm iterates through all pairs of distinct $r, s \in S^{(1)}$ to form $S^{(2)}$ and so on. We call the process of obtaining $S^{(d)}$ from $S^{(d-1)}$ the d-th CHM iteration. Once $S^{(d)} = S^{(d-1)}$, the algorithm terminates.

Example: We illustrate the algorithm for $B = 5$, i.e. with the goal to generate 5-smooth twin integers. The starting set is

$$S^{(0)} = \{1, 2, 3, 4\}.$$

Going through all pairs $(r, s) \in S^{(0)}$ with $r < s$, we see that the only ones that yield a new twin smooth pair $(t, t+1)$ via Equation (4) with t not already in $S^{(0)}$ are $(2,3)$, $(2,4)$ and $(3,4)$, namely,

$$\frac{2}{2+1} \cdot \frac{3+1}{3} = \frac{8}{9}, \quad \frac{2}{2+1} \cdot \frac{4+1}{4} = \frac{5}{6}, \quad \text{and} \quad \frac{3}{3+1} \cdot \frac{4+1}{4} = \frac{15}{16}.$$

Hence, we add 5, 8 and 15 to get the next set as

$$S^{(1)} = \{1, 2, 3, 4, 5, 8, 15\}.$$

The second and third CHM iterations give

$$S^{(2)} = \{1, 2, 3, 4, 5, 8, 9, 15, 24\} \text{ and } S^{(3)} = \{1, 2, 3, 4, 5, 8, 9, 15, 24, 80\}.$$

The fourth iteration does not produce any new numbers, i.e. we have $S^{(4)} = S^{(3)}$, the algorithm terminates here and returns $S^{(3)}$. This is indeed the full set of twin 5-smooth integers as shown in [20], see also [16, Table 1A].

Remark 2. The CHM check that determines whether a pair (r, s) yields an integer solution t to the equation

$$\frac{t}{t+1} = \frac{r}{r+1} \cdot \frac{s+1}{s} \tag{4}$$

can be rephrased by solving this equation for t, which yields

$$t = \frac{r(s+1)}{s-r}. \tag{5}$$

This shows that in order for (r, s) to yield a new pair, $s-r$ must divide $r(s+1)$ and in particular, must be B-smooth as well.

3.2 Generalising the CHM Algorithm

We now present a generalisation of the CHM algorithm, which finds smooth values of any monic quadratic polynomial $f(X) = X^2 + aX + b \in \mathbb{Z}[X] \subseteq \mathbb{Q}[X]$. The algorithm works with elements in the $\mathbb{Q}$-algebra $A = \mathbb{Q}[X]/\langle f(X)\rangle$. Let $\bar{X}$ denote the residue class of X in A. The generalisation closely follows the idea of the CHM algorithm and is based on the observation that for any $r \in \mathbb{Q}$, we have that

$$N_{A/\mathbb{Q}}(r - \bar{X}) = f(r),$$

where $N_{A/\mathbb{Q}}(\alpha)$ denotes the algebraic norm of $\alpha \in A$ over $\mathbb{Q}$. The algorithm now starts with an initial set

$$S^{(0)} = \{r_1 - \bar{X}, \ldots, r_d - \bar{X}\},$$

where r_i are smooth integer values of $f(X)$, which means that the element $r_i - \bar{X}$ has smooth non-zero norm. Next, in the d-th iteration of the algorithm, given any two $\alpha, \beta \in S^{(d-1)}$, compute

$$\alpha \cdot \beta^{-1} \cdot N_{A/\mathbb{Q}}(\beta) = r - s\bar{X}$$

for integers r, s (notice that β is invertible, since it has non-zero norm). Now, if s divides r, we obtain an integer $t = \frac{r}{s}$. It follows that

$$\begin{aligned} f(t) &= N_{A/\mathbb{Q}}\left(\frac{r}{s} - \bar{X}\right) \\ &= N_{A/\mathbb{Q}}(r - s\bar{X})s^{-2} \\ &= N_{A/\mathbb{Q}}(\alpha \cdot \beta^{-1} \cdot N_{A/\mathbb{Q}}(\beta))s^{-2} \\ &= N_{A/\mathbb{Q}}(\alpha)N_{A/\mathbb{Q}}(\beta)s^{-2}. \end{aligned}$$

Since both $N_{A/\mathbb{Q}}(\alpha)$ and $N_{A/\mathbb{Q}}(\beta)$ are B-smooth and s is an integer, it follows that t is a B-smooth value of $f(X)$. The set $S^{(d)}$ is then formed as the union of $S^{(d-1)}$ and the set of all such integral solutions. Finally, we terminate when $S^{(d)} = S^{(d-1)}$.

3.3 Equivalence with Previous Algorithms

We now show that the CHM algorithm, as well as another algorithm by Conrey and Holmstrom [6], are special cases of the generalised algorithm, for the polynomials $f(x) = X^2 + X$, and $f(X) = X^2 + 1$ respectively.

Smooth values of $\boldsymbol{X^2 + X}$. To see that the CHM algorithm (see §3.1) is indeed a special case of the generalised algorithm above, we show how the generalised algorithm works for $f(X) = X(X + 1) = X^2 + X$. Consider the algebra $A = \mathbb{Q}[X]/\langle X^2 + X\rangle$. This embeds into the matrix algebra $M_{2\times 2}(\mathbb{Q})$ via

$$\psi : r + s\bar{X} \to \begin{pmatrix} r & 0 \\ s & r - s \end{pmatrix}.$$

Instead of working with elements in A, we will work with elements in $\psi(A) \subseteq M_{2\times 2}(\mathbb{Q})$ since this simplifies the argument. In this case, for $\alpha \in A$, we have

$$N_{A/\mathbb{Q}}(\alpha) = \det(\psi(\alpha)).$$

The set corresponding to the initial set in the CHM algorithm is

$$S^{(0)} = \{\left(\begin{smallmatrix} 1 & 0 \\ -1 & 2 \end{smallmatrix}\right), \left(\begin{smallmatrix} 2 & 0 \\ -1 & 3 \end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix} B-1 & 0 \\ -1 & B \end{smallmatrix}\right)\}.$$

All these elements clearly have B-smooth norm. The d-th CHM iteration proceeds as follows: For all $\left(\begin{smallmatrix} r & 0 \\ -1 & r+1 \end{smallmatrix}\right), \left(\begin{smallmatrix} s & 0 \\ -1 & s+1 \end{smallmatrix}\right)$ in $S^{(d-1)}$, we try

$$\begin{aligned} \begin{pmatrix} r & 0 \\ -1 & r+1 \end{pmatrix} \begin{pmatrix} s & 0 \\ -1 & s+1 \end{pmatrix}^{-1} s(s+1) &= \begin{pmatrix} r & 0 \\ -1 & r+1 \end{pmatrix} \left(\begin{pmatrix} s+1 & 0 \\ 1 & s \end{pmatrix} \frac{1}{s(s+1)} \right) s(s+1) \\ &= \begin{pmatrix} r(s+1) & 0 \\ -(s-r) & (r+1)s \end{pmatrix}. \end{aligned}$$

Finally, we transform this matrix into the right form, i.e. into a matrix corresponding to an element of the form $\tau = t - \bar{X}$, which means that $\psi(\tau)$ has a -1 in the lower left corner. So, we divide by $s - r$ and end up with the matrix

$$\begin{pmatrix} \frac{r(s+1)}{s-r} & 0 \\ -1 & \frac{(r+1)s}{s-r} \end{pmatrix} = \begin{pmatrix} \frac{r(s+1)}{s-r} & 0 \\ -1 & \frac{r(s+1)}{s-r} + 1 \end{pmatrix}.$$

Now if $\frac{r(s+1)}{s-r}$ is an integer, we add this matrix to the next set $S^{(d+1)}$.

As we have seen in Remark 2, this integer indeed corresponds to the solution (5) of Equation (4) and therefore, the generalised algorithm in the case $f(X) = X^2 + X$ is equivalent to the original CHM algorithm.

Smooth values of $X^2 + 1$. Conrey and Holmstrom later presented a method to generate smooth values of $X^2 + 1$ [6]. Similar to the CHM algorithm, it starts with an initial set $S^{(0)}$ of smooth values of $X^2 + 1$. Again, for $d > 0$ and given $r, s \in S^{(d-1)}, r < s$, they compute

$$\frac{rs - 1}{s + r}.$$

The next set $S^{(d)}$ is then again formed as the union of $S^{(d-1)}$ and the set of all such values that are integers.

It is equally straightforward to verify that this algorithm is also a special case of the generalised CHM algorithm described above in §3.2. We could again work with matrices in $M_{2\times 2}(\mathbb{Q})$, but here, we are actually working in the number field $K = \mathbb{Q}[X]/\langle X^2 + 1\rangle$, which is isomorphic to $\mathbb{Q}(i)$, where $i^2 = -1$. The product of the elements $\alpha = r - i$ and $\beta = s - i$ is given as

$$\alpha\beta = (r - i)(s - i) = (rs - 1) - (r + s)i.$$

Conrey and Holmstrom's method then simply tries all such products $\alpha\beta$. However, a possibly better choice could be to use

$$\alpha\beta^{-1} N_{K/\mathbb{Q}}(\beta) = \alpha\bar{\beta} = (r - i)(s + i) = (rs + 1) - (s - r)i$$

as described in our generalisation. This is due to the fact that

$$\frac{rs + 1}{s - r}$$

is more likely to be an integer, which means that we can expect the algorithm to converge faster.

Whichever option is chosen, one tries to divide by $r+s$ resp. $s-r$, and if the result is an element in $\mathbb{Z}[i]$, it is added to the next set $S^{(d)}$ of smooth values of X^2+1. Conrey and Holmstrom's method is therefore another special case of the generalised algorithm.

Remark 3. We note that neither the generalised CHM algorithm, nor any of the previous special cases give any guarantees to what proportion of B-smooth values of $f(X)$ it finds. However, for the previous special case algorithms, certain conjectural results have been stated, based on numerical evidence, which suggests that the algorithm returns all but a small fraction of all smooth values of the respective quadratic polynomials. We make no similar claims for the general case algorithm.

4 Searching for Large Twin Smooth Instances: CHM in Practice

Ideally, the CHM algorithm could be run as described in [7] with a large enough smoothness bound B to find twin smooths of cryptographic sizes. However, experiments suggest that this is not feasible in practice. We report on data obtained from an implementation of the pure CHM algorithm in §4.1, present several optimisations in §4.2 and details on our optimised implementation in §4.3.

4.1 Running CHM in Practice

In order to collect data and assess the feasibility of finding large enough twin smooths, we implemented a somewhat optimised version of the pure CHM algorithm. In particular, this implementation is parallelised, and avoids multiple checks of the same pairs of twin smooths (r, s). Furthermore, we iterate through smoothness bounds: We start with a small bound B_1 and the initial set $S_1^{(0)} = \{1, \ldots, B_1 - 1\}$, and use the CHM algorithm to iteratively compute sets $S_1^{(i)}$ until we reach some d_1 such that $S_1^{(d_1)} = S_1^{(d_1-1)}$. In the next iteration, we increase the smoothness bound to $B_2 > B_1$, define the initial set $S_2^{(0)} = S_1^{(d_1)} \cup \{B_1, \ldots, B_2 - 1\}$. Again we compute CHM iterations until we find d_2 such that $S_2^{(d_2)} = S_2^{(d_2-1)}$, where we avoid checking pairs (r, s) that have been processed in earlier iterations. Ideally, we could repeat this procedure until we reach a smoothness bound B_i for which the CHM algorithm produces large enough twin smooths for cryptographic purposes. However, our data suggests that this is infeasible in practice due to both runtime and memory limitations.

In particular, we ran this approach up to the smoothness bound $B = 547$, and extrapolating the results gives us rough estimations of the largest possible pair and number of twin smooths per smoothness bound.

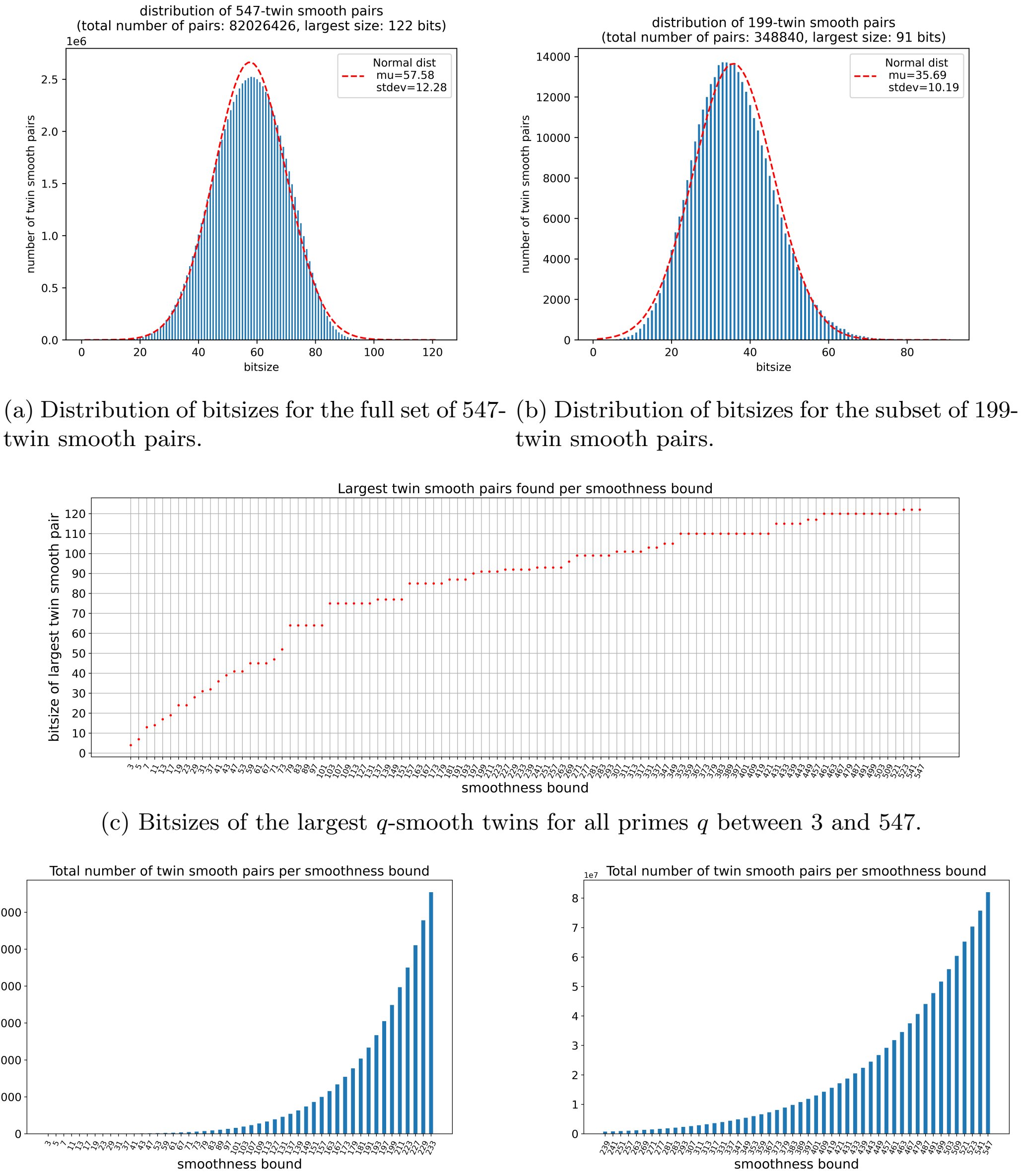
After the $B = 547$ iteration, the set of twin smooths contains 82,026,426 pairs, whose bitlength distribution roughly equals a normal distribution centered around 58-bit. The largest pair has a bitlength of 122 bits. An evaluation of the obtained set is shown in Figure 1. Figure 1a shows the distribution of bitsizes in the full set, while Figure 1b shows that of the subset of all 199-smooth twins obtained in this run. Figure 1c shows the bitsize of the largest q-smooth twin pairs for each prime q between 3 and 547. And Figures 1d and 1e show the number of q-smooth twins for each such q.

Using the data of these experiments, we can attempt to estimate at which smoothness bound B we might expect to reach twin smooths of cryptographic sizes with this approach, i.e., of at least 256-bit, and the memory required to run iterations to reach this B.

The data visualised in Figure 1c allows to interpolate a function describing the smoothness bound necessary such that the largest twin smooth pair obtained when running CHM with this bound has a specific bit size. Using a curve fitting of a cubic polynomial to extrapolate to reach a 256-bit pair indicates that the required smoothness bound is larger than 5,000. A similar interpolation of the data displayed in Figures 1d and 1e suggests that the set of B-smooth twins for this B is roughly 2^{49}. Given that the effort for CHM iterations grows quadratically with the set size, these estimates indicate that it is not feasible to reach cryptographically sized smooth twins with the original CHM algorithm.

4.2 Optimisations

One major issue with running the plain CHM algorithm for increasing smoothness bound is the sheer size of data that needs to be dealt with. The sets $S_i^{(d_i)}$ grow very rapidly and the quadratic complexity of checking all possible pairs (r, s) leads to a large runtime. The natural question that arises is whether CHM can be restricted to checking only a certain subset of such pairs without losing any or too many of the new smooth neighbors. Furthermore, if the purpose of running the CHM algorithm is not to enumerate all twin smooth pairs for a given smoothness bound but



(a) Distribution of bitsizes for the full set of 547-twin smooth pairs.

(b) Distribution of bitsizes for the subset of 199-twin smooth pairs.

(c) Bitsizes of the largest q-smooth twins for all primes q between 3 and 547.

(d) Number of q-smooth twins for all primes q between 3 and 233.

(e) Number of q-smooth twins for all primes q between 239 and 547.

Fig. 1: Evaluation of the set of 547-smooth twins obtained by running the original CHM algorithm with smoothness bound $B = 547$. The bitsize of a pair $(r, r+1)$ is $\lfloor \log r \rfloor + 1$. Data for the number of q-smooth twins for all primes q up to 547 has been split into two histograms of different scale.

instead, to produce a certain number of pairs of a given size or to obtain some of the largest pairs, it might even be permissible to omit a fraction of pairs.

To find a sensible way to restrict to a smaller set, we next discuss which pairs (r, s), $r < s$ result in a given twin smooth pair $(t, t + 1)$ via

$$\frac{r}{r+1} \cdot \frac{s+1}{s} = \frac{t}{t+1}. \tag{6}$$

This is discussed in [7, §3], but we elaborate on it in a slightly different way here. Let $t > 0$, let u be any divisor of t and v any divisor of $t + 1$. Let $h, x \in \mathbb{Z}$ be given by $t = uh$ and $t + 1 = vx$ (where $u, v, h, x > 0$). Therefore, $v/u = h/x + 1/(ux)$. If $u < v$ then $h > x$ and if $u > v$ then $h < x$. We therefore fix $u < v$ (otherwise switch the roles of u, v and h, x). Since $u < v$, the pair

$$(r, s) = (t - \frac{u}{v}(t + 1),\ \frac{v}{u}t - (t + 1) = \frac{v}{u}r) \tag{7}$$

satisfies Equation (6) and it follows that

$$r = u(h - x),\ r + 1 = x(v - u),\ s = v(h - x),\ s + 1 = h(v - u). \tag{8}$$

Therefore, $s/r = v/u$ and $(s + 1)/(r + 1) = h/x$, $u < v$, $h > x$ and $0 < r < s$. This also means that $s = r + (v - u)(h - x)$, $t = r + ux$ and that $\gcd(r(s + 1), s(r + 1)) = s - r = (v - u)(h - x)$ (note that $\gcd(uh, vx) = \gcd(t, t + 1) = 1$).

Conversely, given (r, s) with $r > 0$ that satisfy Equation 6, define $u = r/\gcd(r, s)$ and $v = s/\gcd(r, s)$, then $s > r$, $u < v$ and $u \mid t$, $v \mid (t + 1)$. Hence we have the correspondence between the set of pairs (r, s) with $r < s$ that yield a new twin pair $(t, t + 1)$ via Equation (6) and the set of pairs of divisors of t and $t + 1$ described in [7, §3] as follows:

$$\begin{aligned} &\{(r, s) \mid r < s \text{ and } r(s + 1)(t + 1) = s(r + 1)t\} \\ &\quad\longleftrightarrow \{(u, v) \mid u < v \text{ and } u \mid t,\ v \mid (t + 1)\}. \end{aligned} \tag{9}$$

However, this correspondence does not identify the pairs (r, s) corresponding to twin smooths, i.e. given (u, v) there is no guarantee that any of $r, r + 1, s, s + 1$ are B-smooth. This is not discussed in [7, §3]. The next lemma fills this gap by stating an explicit condition on the divisors u, v, h, x.

Lemma 1. *Let $t \in \mathbb{Z}$ such that $t(t + 1)$ is B-smooth. Let (u, v) be a pair of divisors such that $t = uh$, $t + 1 = vx$ and let (r, s) be defined as in Equation (7).*

Then $r(r + 1)s(s + 1)$ is B-smooth if and only if $(v - u)(h - x) = s - r$ is B-smooth.

Proof. As divisors of t and $t + 1$, u and v as well as h and x are all B-smooth. The statement follows from the Equations (8). □

Using similar sized pairs. We next consider the following condition to restrict the visited pairs (r, s) in CHM as a mechanism to reduce the set size and runtime. Let $k > 1$ be a constant parameter. We then only check pairs (r, s) if they satisfy

$$0 < r < s < kr. \tag{10}$$

Assume that (r, s) results in a pair $(t, t + 1)$ through satisfying Equation (6). As seen above, $\frac{s}{r} = \frac{v}{u}$ for $u \mid t$, $v \mid (t + 1)$, so we can use (u, v) to determine which values k are useful. Since $\frac{v}{u} < k$, it follows $s = \frac{v}{u}t - (t + 1) < (k - 1)t$. If we are only interested in obtaining a new t from a pair (r, s) such that $s < t$, we can take $k \leq 2$, overall resulting in $1 < k \leq 2$.

This k seems to be a good quantity to study as we can relate it to the factors of $v - u$. Indeed, $v - u = u(\frac{v}{u} - 1) = u(\frac{s}{r} - 1)$ and we have $s < kr$.

Definition 2. *Let $(r, r + 1)$ and $(s, s + 1)$ be twin smooths with $r < s$ and $k \in \mathbb{R}$ with $1 < k \leq 2$. We call the pair (r, s) k-*balanced *if $r < s < k \cdot r$.*

We want to find a k such that a k-balanced pair (u, v) subject to the above conditions will yield a balanced r, s such that $r, r+1, s, s+1$ are B-smooth, or equivalently that $v-u$ and $h-x$ are.

Running the CHM algorithm only with 2-balanced pairs (r, s) then guarantees that any t produced by Equation 6 will be larger than the inputs r and s. Although we sacrifice completeness of the set of twin B-smooths with this approach, we can significantly reduce the runtime.

We can even push this approach further. Recall that we require $\gcd(r(s+1),(r+1)s) = s-r$ in order to generate a new pair of twin smooths $(t, t+1)$. By Lemma 1, this also implies that $\varDelta = s-r$ must be B-smooth. Heuristically, the smaller $\varDelta$ is, the better the chances for $\varDelta$ to be smooth. Furthermore, if $\varDelta$ contains small and only few prime factors, the probability for the condition $\varDelta = \gcd(r(s+1),(r+1)s)$ is relatively high. We can summarise this in the following heuristic.

Heuristic 1 *Let $k_1, k_2 \in \mathbb{R}$ with $1 < k_1 < k_2 \leq 2$, and (r_1, s_1) resp. (r_2, s_2) a k_1- resp. k_2-balanced pair of twin smooths. Then the probability for (r_1, s_1) to generate new twin smooths via the CHM equation is larger than that for (r_2, s_2).*

In order to save additional runtime, we can thus pick k closer to 1, and only check the pairs (r, s) that are most likely to generate new twin smooths. Therefore, we can still expect to find a significant portion of all twin B-smooths for a given smoothness bound B. We expand on the choice of k and different ways of implementing this approach in §4.3.

Thinning out between iterations. Another approach to reduce both runtime and memory requirement is to thin out the set of twin smooths between iterations. In particular, once we finished all CHM steps for a certain smoothness bound B_i, we can remove twins from the set $S_i^{(d_i)}$ based on their likeliness to produce new twin smooths before moving to the next iteration for B_{i+1}.

One possible condition for removing twins is to look at their smoothness bounds. Let $(r, r+1)$ be B_1-smooth, $(s, s+1)$ be B_2-smooth (but not B-smooth for any $B < B_2$), and $B_1 \ll B_2$. Since $(s, s+1)$ contains (multiple) prime factors larger than B_1, they cannot be contained in $(r, r+1)$, which makes the requirement $\gcd(r(s+1),(r+1)s) = s-r$ heuristically less likely to be satisfied. However, in practice it turns out that the differences between the smoothness bounds we are concerned with are not large enough for this heuristic to become effective.

In our experiments, it turned out to be more successful to keep track of how many new twin smooths each r produces. We can then fix some bound m, and discard twins that produced less then m twins after a certain number of iterations. Our experiments suggest that using this approach with carefully chosen parameters yields a noticeable speedup, but fails completely at reducing the memory requirements, as we still need to keep track of the twins we already found. Furthermore, in practice the approach of only using k-balanced twins turned out to be superior, and hence we focus on this optimisation in the following.

4.3 Implementation

We implemented the CHM algorithm with several of the aforementioned optimisations in C++, exploiting the fact that it parallelises perfectly. Note that some of our approaches require the set of twin smooths to be sorted with respect to their size. Hence, an ordered data structure is used for storing the twins set. We used the following techniques and optimisations.

CHM step. For each pair (r, s) considered by the implementation, we have to check if Equation (6) holds. As mentioned in §4.2, this requires that $\gcd(r(s+1),(r+1)s) = s-r$ is satisfied. However, we can completely avoid the gcd calculation by observing that we require $r \cdot (s+1) \equiv 0 \mod (s-r)$. Only if this is the case we perform a division to compute t, which represents the new pair of twin smooths $(t, t+1)$. Therefore, we only perform one modular reduction per considered pair (r, s), followed by one division if the CHM step is successful. This is significantly cheaper than a naïve implementation of Equation (6) or a gcd computation.

Data structure. Initially the set of twins was organised in a standard C array, that each time an iteration completed was reallocated to increase its size, and reordered.

To avoid the overall inefficiency of this method we moved to use the C++ standard library std::set. This data structure is implemented with a Red Black tree, guarantees $O(\log N)$ insertion and search, while keeping the elements always ordered.

We then moved to use B+Trees [3], that have the same guarantees for insertion, search, and ordering, but are more efficient in the memory usage. Because the elements of a B+Tree are stored close to each other in memory it becomes much faster to iterate through the set, an operation that is necessary for creating the pairs used in each computation.

Implemented optimisations. As discussed in §4.2, we focus on the case of k-balanced pairs (r, s), which satisfy $r < s < k \cdot r$. Compared to the full CHM algorithm, this leads to a smaller set of twin smooths, but allows for much faster running times. We implemented the k-balanced approach in various different flavours.

Global-k. In the simplest version - the `global-`k approach - we initially pick some k with $1 < k \leq 2$, and restrict the CHM algorithm to only check k-balanced pairs (r, s). The choice of k is a subtle manner: Picking k too close to 1 may lead to too many missed twin smooths, such that we cannot produce any meaningful results. On the other hand, picking k close to 2 may result in a relatively small speedup, which does not allow for running CHM for large enough smoothness bounds B. Unfortunately, there seems to be no theoretical handle on the optimal choice of k, which means that it has to be determined experimentally. We note that when picking an aggressive bound factor $k \approx 1$, small numbers r in the set of twins S may not have any suitable $s \in S$ they can be checked with. Thus, we pick a different bound, e.g. $k = 2$, for numbers below a certain bound, e.g. for $r \leq 2^{20}$.

Iterative-k. Instead of iterating through smoothness bounds B_i as described in §4.1 and using the `global-`k approach, we can switch the roles of B and k if we are interested in running CHM for a fixed smoothness bound B. We define some initial value k_0, a target value $k_{\max}$, and a step size $k_{\text{step}} > 0$. In the first iteration, we run CHM as in the `global-`k approach, using k_0. The next iteration then increases to $k_1 = k_0 + k_{\text{step}}$, and we add the condition to not check pairs (r, s) if they were already checked in previous iterations. We repeat this iteration step several times until we reach $k_{\max}$. Compared to the `global-`k approach, this allows us to generate larger B-smooth twins faster, since we restrict to the pairs (r, s) first that are most likely to generate new twins. However, the additional checks if previous pairs have been processed in earlier iterations add a significant runtime overhead. Thus, this method is more suitable for finding well-suited choices of k, while actual CHM searches benefit from switching to the `global-`k approach.

Constant-range. In both the `global-`k and `iterative-`k approach, the checks if a pair (r, s) is k-balanced, or has been processed in earlier iterations, consumes a significant part of the overall runtime. Therefore, we can use constant ranges to completely avoid these checks. Since we always keep the set of twins S sorted by size, the numbers s closest to r (with $s > r$) are its neighbors in S. Thus, we can sacrifice the exactness of the k-balanced approaches above, and instead fix a range R and for each r check (r, s) with the R successors s of r in S. As shown below, this method significantly outperforms the `global-`k approach due to the elimination of all checks for k-balance. This is true even when R is large enough to check more pairs than are considered in the `global-`k approach for a given k.

Variable-range. Similar to the `constant-range` approach, we can adapt the range R depending on the size of r. For instance, choosing r at the peak of the size distribution will lead to many possible choices of s such that (r,s) are balanced. Hence, we can choose a larger range R whenever more potential pairs exist, while decreasing R otherwise. In practice, the performance of this method ranks between `global-`k and `constant-range` by creating roughly the same pairs that `global-`k

Variant	**Parameter**	**Runtime**	**Speedup**	**#twins**	**#twins from largest 100**
Full CHM	-	4705s	1	2300724	100
`global-`k	$k = 2.0$	364s	13	2289000	86
	$k = 1.5$	226s	21	2282741	82
	$k = 1.05$	27s	174	2206656	65
`constant-range`	$R = 10000$	82s	57	2273197	93
	$R = 5000$	35s	134	2247121	87
	$R = 1000$	16s	294	2074530	75

Table 1: Performance results for different variants of our CHM implementation for smoothness bound $B = 300$. Speedup factors refer to the full CHM variant.

creates without any of the overhead of the balance checks. If R is chosen large enough such that the `constant-range` approach ends up generating more pairs than `global-`k, then `variable-range` performs better. Realistically, the size of the range R increases by (very) roughly 3% for each prime number smaller than the smoothness bound B, and slows down the algorithm drastically at higher smoothness, similarly to the k-based approaches.

Remark 4. Similar to the `variable-range` approach, we experimented with a variant of the `global-`k approach, which adjusts k according to the size of r to find suitable s for the CHM step. However, the `constant-range` and `variable-range` approaches turned out to be superior in terms of performance, and therefore we discarded this `variable-`k variant.

Performance comparison. In order to compare the implications of the optimisations in practice, we ran different variants of the CHM implementation for the fixed smoothness bound $B = 300$. All experiments ran on a machine configured with 4 x Xeon E7-4870v2 15C 2.3 GHz, 3072 GB of RAM. The total amount of parallel threads available was 120. As described above, the `global-`k and `constant-range` approach significantly outperform their respective variants, hence we focus on different configurations of these two methods.

The results are summarised in Table 1. For both the `global-`k and the `constant-range` approach we measured the results for conservative and more aggressive instantiations, where smaller values of k and R are considered more aggressive. It is evident that already for the conservative instantiations, we gain significant performance speedup, while still finding almost the full set of twin smooths, and most of the 100 largest 300-smooth twins. For the more aggressive instantiations, we miss more twins, yet still find a significant amount of large twins.

As discussed above, the `constant-range` approach outperforms the `global-`k approach in terms of runtime, due to the elimination of all checks for k-balance of twins. Interestingly, while very aggressive instantiations of `constant-range` miss more twin smooths, they find a larger share of the largest 100 twins than their `global-`k counterpart. Therefore, we conclude that for larger smoothness bounds B, for which we cannot hope to complete the full CHM algorithm, `constant-range` is the most promising approach for obtaining larger twin smooths within feasible runtimes.

Remark 5. While all optimisations lose a small proportion of the largest twin smooths, they are not necessarily lost permanently. In practice, when iterating to larger smoothness bounds B_i, we often also find some B_j-smooth twins for bounds $B_j < B_i$. Thus, the size of the set of 300-smooth twins usually increases in the optimised variants when moving to larger B.

Remark 6. In the following sections, we will require twin smooths of a certain (relatively small) bitlength. This can easily be incorporated into all implemented variants by removing all twins above this bound after each iteration. This means that we cut off the algorithm at this size, and do not attempt to obtain larger twins, which significantly improves the runtime and memory requirements.

n	$p_n(x)^2 - 1$
2	$4x^2(x-1)(x+1)$
3	$4x^3(x-1)(x^2+x+1)$
4	$4x^4(x-1)(x+1)(x^2+1)$
5	$4x^5(x-1)(x^4+x^3+x^2+x+1)$
6	$4x^6(x-1)(x+1)(x^2-x+1)(x^2+x+1)$

Table 2: Factorisation of $p_n(x)^2 - 1$ for $n = 2, 3, 4, 5, 6$, where $p_n(x) = 2x^n - 1$

5 Fantastic p's and where to find them: Cryptographic Primes of the form $p = 2r^n - 1$

This section focuses on finding primes suitable for isogeny-based cryptographic applications. As discussed in the previous sections, the pure CHM method does not allow for us to directly compute twins of at least 256 bits as required for this aim. However, some cryptographic applications, for example the isogeny-based signature scheme SQISign, do not need twins $(r, r+1)$ that are fully smooth. Indeed, the current incarnation of SQISign requires a prime p that satisfies $2^f T \mid p^2 - 1$, where f is as large as possible, and $T \approx p^{5/4}$ is smooth and odd [13]. This flexibility allows us to move away from solely using CHM and, instead, to use CHM results as inputs to known methods for finding such primes. At a high level, we will find fully smooth twins of a smaller bit-size via CHM and boost them up using the polynomials $p_n(x) = 2x^n - 1$ (for carefully chosen n). Hence, if $r, r+1$ are fully smooth integers and n is not too large, we can guarantee a large proportion of $p_n(r)^2 - 1$ to be smooth.

Notation. For a variable x, we will denote $2x^n - 1$ by $p_n(x)$, and the evaluated polynomial $p_n(r)$ by p, emphasizing that it is an integer.

General method. In this section, we will give a more in-depth description of the approach to obtaining cryptographic sized primes p, such that $p^2 - 1$ has $\log T'$ bits of B-smoothness, where $T' = 2^f T$. We recall that for our SQISign application, we have $\log p \in \{256, 384, 512\}$ for NIST Level I, III and V (respectively), $T \approx p^{5/4}$ and f as large as possible. In the current implementation of SQISign, $f \approx \lfloor \log\left(p^{1/4}\right) \rfloor$ (i.e., $T' \approx p^{3/2}$), and therefore, we aim for this when finding primes.

Fix a smoothness bound B and let $p_n(x) = 2x^n - 1$. We have $p_n(x)^2 - 1 = 2x^n(x-1)f_n(x)$, where the factorisation of $f_n(x)$ for $n = 2, \ldots, 6$ is given in Table 2.

We observe that for n even, both $x+1$ and $x-1$ appear in the factorisation of $p_n(x) - 1$. In this case, for twin smooths $(r, r \pm 1)$, evaluating $p_n(x)$ at r guarantees that we have a smooth factor $4x^n(x \pm 1)$ in $p^2 - 1$. For n odd, we will only have that $x - 1$ appears in the factorisation, and therefore only consider twins $(r, r-1)$ to guarantee we have B-smooth factor $4x^n(x-1)$.

The first step is to use our implementation of the CHM algorithm, described in Sections 3 and 4, to obtain B-smooth twins $(r, r \pm 1)$ of bitsize approximately $(\log p - 1)/n$. We then obtain primes of suitable sizes via computing $p = p_n(r)$ for all candidate r, as described above. By construction, $p^2 - 1$ has guaranteed $\frac{n+1}{n}(\log(p) - 1) + 2$ bits of smoothness. We then require that the remaining factors have at least

$$\max\left(0,\ \frac{3}{2}\log p - \left(\frac{n+1}{n}\left(\log p - 1\right) + 2\right)\right)$$

bits of B-smoothness. In Section 5.2, we will discuss the probability obtaining this smoothness from the remaining factors.

5.1 Choosing n

For small n, we require CHM to find twin smooths of *large* bit size. For certain bit sizes, running full CHM may be computationally out of reach, and therefore we use a variant that may not find all twins. In this case, however, we have more guaranteed smoothness in p^2-1 and so it is more likely that the remaining factors will have the required smoothness. For large n, we can obtain more twin smooths from CHM (in some cases, we can even exhaustively search for all twin smooths), however we have less guaranteed smoothness in p^2-1. Finding values of n that balance these two factors will be the focus of this section.

***n* = 2.** Let $(r, r \pm 1)$ be twin smooth integers and let $p = 2r^2 - 1$. In this case, $2r^2(r \pm 1) \mid T'$, meaning that $\log T' \geq \frac{3}{2} \log p$, and we have all the required smoothness. Write $T' = 2^f T = 2r^2(r \pm 1)$ where T is odd. If $f < \lfloor \log\left(p^{1/4}\right) \rfloor$, we have $T > p^{5/4}$, and we do not have to rely on a large power of 2 dividing $r-1$. Otherwise, we turn to Section 5.2 to estimate the probability of $r \mp 1$ having enough small factors to make up for this difference.

Suppose we target primes with λ bits of classical security, i.e., we need a prime of order $p \approx 2^{2\lambda}$. For $n = 2$, this corresponds to finding twin smooths of size $\approx 2^{\lambda - \frac{1}{2}}$, and so is only suitable for finding NIST Level I parameters due to the limitations of the CHM method (see Section 4). One could instead use other techniques for finding large enough twins for $n = 2$, such as the PTE sieve [9], at the cost of significantly larger smoothness bounds. Alternatively, we can move to higher n, which comes at the cost of loosing guaranteed smoothness. Another challenge here is that, given the relatively large size of the twins, it appears difficult to find enough twins for obtaining primes with a large power of two.

***n* = 3.** Let $(r, r-1)$ be twin smooth integers and let $p = 2r^3 - 1$. Here, we can guarantee that the smooth factor T' of $p^2 - 1$ is at least of size $\approx p^{4/3}$. If $f < \lfloor \log_2\left(p^{1/12}\right) \rfloor$, we have $T > p^{5/4}$. Otherwise, we require that there are enough smooth factors in $r^2 + r + 1$ to reach this requirement.

Here, for λ bits of classical security, we need to target twin smooth integers of size $\approx 2^{\frac{2\lambda-1}{3}}$. In this case, the CHM method will (heuristically) allow us to reach both NIST Level I and III parameters.

***n* = 4.** Let $(r, r \pm 1)$ be twin smooth integers and $p = 2r^4 - 1$. Here we can only guarantee a factor of size $\approx p^{5/4}$ of $p^2 - 1$ to be smooth. When accounting for the power of two, we must hope for other smooth factors. As $p_n(x) - 1$ splits into (relatively) small degree factors, namely $p_n(x) - 1 = 2(x-1)(x+1)(x^2+1)$, the probability of having enough B-smooth factors is greater (than if there was, for example, a cubic factor).

In contrast to the previous cases, this setting should be suitable for targeting all necessary security parameters. However, for the NIST Level I setting, the work by De Feo, Leroux and Wesolowski [13][§5.2] showed that the best one could hope for here while maximising the power of two gives SQISign parameters with a smoothness bound of ≈ 1800. While this is a better smoothness bound than the NIST Level I prime with the best performance for SQISign, it does not perform as well in practice. Indeed, most of the odd primes less than 1800 that appear in $p^2 - 1$ are relatively large, making isogeny computation relatively slow. In the best performing prime, however, a large power of 3 divides $p^2 - 1$, and most of its other odd prime divisors are fairly small.

Other *n*. For larger n, the amount of guaranteed smoothness decreases, and thus the probability that the remaining factors have the required smoothness is small. Indeed, we find that only $n = 6$ has the correct balance of requiring small twin smooths while still having a reasonable probability of success. This is primarily due to the factorisation of $p_6(x) - 1 = 2(x-1)(x+1)(x^2-x+1)(x^2+x+1)$, having factors of degree at most 2, which improves the probability that we have enough smooth factors. In contrast, $n = 5$ results in more guaranteed smoothness than $n = 6$, but requires the quartic factor in $p_5(x) - 1$ to provide the necessary smoothness, which is relatively unlikely.

While one could use $n = 6$ to find NIST Level I parameters, this larger n shines in its ability to give us both NIST Level III and V parameters.

	n	$\log_2(r)$	**Probability of B-smooth** $(r, r \pm 1)$	**Probability of $p^2 - 1$** $\log T'$**-bits B-smooth given** $(r, r \pm 1)$ **twin smooth**	**Extra Smoothness Needed**
NIST-I	2	≈ 127.5	$2^{-58.5}$	1	0
$B = 2^9$	3	≈ 85.0	$2^{-32.1}$	$2^{-12.1}$	42
$\log p = 256$	4	≈ 63.8	$2^{-20.5}$	$\approx 2^{-22.4}$	63.3
$\log T' = 384$	6	≈ 42.5	$2^{-10.4}$	$\approx 2^{-32.2}$	84.5
NIST-III	2	≈ 191.5	$2^{-55.7}$	1	0
$B = 2^{14}$	3	≈ 127.7	$2^{-30.5}$	$2^{-11.7}$	63.3
$\log p = 384$	4	≈ 95.8	$2^{-19.4}$	$\approx 2^{-15.7}$	95.3
$\log T' = 576$	6	≈ 63.8	$2^{-9.7}$	$\approx 2^{-19.2}$	127.2
NIST-V	2	≈ 255.5	$2^{-63.7}$	1	0
$B = 2^{17}$	3	≈ 170.3	$2^{-35.2}$	$2^{-13.5}$	84.7
$\log p = 512$	4	≈ 127.8	$2^{-22.6}$	$\approx 2^{-18.2}$	127.3
$\log T' = 768$	6	≈ 85.2	$2^{-11.5}$	$\approx 2^{-22.5}$	169.8

Table 3: Assuming that $(r, r \pm 1)$ are twin smooth integers and p has $\log p$ bits, calculates the probability of having a B-smooth divisor $T' \mid p^2 - 1$ of size $\approx p^{3/2}$. More details in text.

5.2 Probability of Sufficient Smoothness

In this section, we determine the probability of obtaining cryptographic primes with sufficient smoothness using the methods outlined above.

First, we determine the probability of finding fully B-smooth twins $(r, r \pm 1)$ using the counting function Ψ, defined in §2.2.

To calculate the probability of $p - 1$ having $\log T'$-bits of B-smoothness, given that the factor $r(r \pm 1) \mid p - 1$ is already fully smooth, we will use the counting function Θ studied by Banks–Shparlinski and introduced in §2.2. Indeed, as we only require around $\log T'$-bits of smoothness from $p - 1$, it may only be partially smooth. As discussed in the section above, we restrict to $n = 2, 3, 4, 6$.

Recall that $p_n(x)^2 - 1 = 2x^n(x-1)f(x)$, where, in general, $f(x)$ is not irreducible (see Table 2). Write $f(x) = f_1(x) \cdots f_k(x)$, where each f_i is irreducible of degree $d_i = \deg(f_i)$ and $d = \deg(f)$. To calculate the probabilities, we require that the probability of $f(x)$ having at least $\log_2 D$-bits of B-smoothness is the product of the probabilities of each of its factors f_i having at least $\log_2 D_i$-bits of B-smoothness where $\log_2 D = \sum_{i=1}^{k} \log_2 D_i$. We can view this as an extension of [9, Heuristic 1]. Note that the only constraint on how the smoothness is distributed between the factors $f_i(x)$ is that the total bit size of B-smooth factors must equal $\log_2 D$. We could, for example, sum over all the possible distributions of smoothness using the inclusion-exclusion principle. However, in distributions where one of the factors has a very small amount of smoothness, we fall out of the ranges allowed as input into Θ determined by Theorem 1. Therefore, for simplicity, we will assume that smoothness is distributed evenly between the remaining factors (weighted by the degree), i.e., $\log_2 D_i = (d_i \log_2 D)/d$. In reality, this only gives us a lower bound for the probability, but this will suffice for our purposes. Obtaining a more theoretical and accurate grasp on these probabilities is left as an avenue for future research.

In Table 3, we give an overview of the relevant probabilities for NIST Level I, III, and V parameters, calculated as described above.

6 Results and Comparisons

In this section we give the concrete results that were obtained from our experiments with the CHM algorithm, and analyse the various twins that we need for SQISign in accordance with the relevant bitsizes mentioned in Table 3.

6.1 Record Twin Smooth Computations

We ran the optimised full CHM algorithm with $B = 547$ and found a total of 82,026,426 pairs of B-smooth twins. For the application mentioned in the previous section, we only need twins of a certain bitsize. Within this set of twins, 9,218,648 pairs $(r, r+1)$ fall in the range $2^{60} < r < 2^{64}$; 1,064,249 pairs fall in the range $2^{81} < r < 2^{85}$; 31,994 pairs fall in the range $2^{92} < r < 2^{96}$; and, only 1 pair falls in the range $2^{120} < r < 2^{128}$. This pair in the final interval is the largest pair found in this run, with $r = 4012031241848866526424165796047749375$, and factorisations:

$$\begin{aligned} r &= 5^4 \cdot 7 \cdot 13^2 \cdot 17^2 \cdot 19 \cdot 29 \cdot 41 \cdot 109 \cdot 163 \cdot 173 \cdot 239 \cdot 241^2 \cdot 271 \cdot 283 \\ &\quad \cdot 499 \cdot 509, \text{ and} \\ r + 1 &= 2^8 \cdot 3^2 \cdot 31^2 \cdot 43^2 \cdot 47^2 \cdot 83^2 \cdot 103^2 \cdot 311^2 \cdot 479^2 \cdot 523^2. \end{aligned}$$

As we will see later, the number of 64-bit and 85-bit twins we found in this run is enough to find attractive parameters for SQISign. The 96-bit twins will give us parameters with the required smoothness, however we do not have enough pairs to hope to find a prime p where $p^2 - 1$ is divisible by a large power of two.

Table 3 shows that finding many twins of around 128 bits in size is likely to be fruitful in the search for SQISign-friendly parameters, so we ran the algorithm for $B = 1300$ using the `constant-range` optimisation with a range $R = 5000$, in order to specifically target twins $(r, r+1)$ with $r > 2^{115}$. In this run we found 1,091 such pairs - the largest of these pairs is the following 145-bit twin $(r, r+1)$ with $r = 36132012096025817587153962195378848686084640$, where

$$\begin{aligned} r &= 2^5 \cdot 5 \cdot 7 \cdot 11^2 \cdot 13 \cdot 23 \cdot 53 \cdot 71 \cdot 109 \cdot 127 \cdot 131 \cdot 193 \cdot 251 \cdot 283 \cdot 307 \\ &\quad \cdot 359 \cdot 367 \cdot 461 \cdot 613 \cdot 653 \cdot 1277, \text{ and} \\ r + 1 &= 3^2 \cdot 29^2 \cdot 31^2 \cdot 43^2 \cdot 59^2 \cdot 61^2 \cdot 73^2 \cdot 79^2 \cdot 89^2 \cdot 167^2 \cdot 401^2 \cdot 419^2. \end{aligned}$$

Among the 1,091 twins CHM found, 184 pairs fall in the range $2^{120} < r < 2^{128}$, which was sufficient to find some SQISign-friendly parameters (though not at all NIST security levels).

In addition, we also ran CHM with $B = 2^{11}$ to obtain a large number of twin smooth integers in the range $2^{55} < r < 2^{100}$ (see Remark 6 in the setting where we want to find twins in such an interval). This run was performed using the `constant-range` optimisation with a range $R = 2500$, and produced 608,233,761 pairs of twins lying in this range. Compared with the $B = 547$ run, the yield from this run gave ample twins with $2^{92} < r < 2^{96}$, which was sufficient to find SQISign parameters with the desirable large power of two.

6.2 Concrete Parameters for SQISign

We now turn to giving a list of SQISign-friendly primes that target NIST Level I, III, and V. Recall from Section 1 that this means that we need to find primes p with $2^f \cdot T \mid p^2 - 1$. We need the exponent f to be as large as possible and the cofactor $T \approx p^{5/4}$ to be B-smooth, aiming to keep the ratio $\sqrt{B}/f$ as small as possible; this quantity is a rough cost metric for the performance of the signing algorithm in SQISign [13, §5.1]. To complement this, the exponent f controls the performance of the verification of SQISign; the larger this exponent is the faster the verification is. We may run into circumstances where the signing cost metric is minimised, but the power of two is not large enough or vice-versa. We aim to balance these as much as possible, thus finding parameters that maximise the power of two while minimising the signing cost metric.

Though we need $T \approx p^{5/4}$, if this cofactor is too close to $p^{5/4}$, then the underlying heuristics within the generalised KLPT algorithm might fail and one cannot guarantee a successful signature in SQISign [13, §3.2]. Thus, in practice we need $T \approx p^{5/4+\epsilon}$ for some small ϵ (e.g., $0.02 < \epsilon < 0.1$).

We find parameters for NIST Level I, III and V by searching for 256, 384 and 512-bit primes, respectively. For those primes targeting the higher security levels, these are the first credible SQISign-friendly primes. In what follows, we look at each security level and analyse the most noteworthy primes found in our searches. When stating the factorisations of $p \pm 1$ for the mentioned primes, the factors in grey are the rough factors which are not needed for SQISign. A full collection of our best SQISign-friendly primes is showcased in Table 4.

NIST-I parameters. We targeted 256-bit primes using $n = 2$ and 3. Given that our CHM runs produced a lot more twins at the 85-bit level than at the 128-bit level, we expect to find more primes using $n = 3$, which was indeed the case. It is worth noting that some primes found with $n = 2$ gave rise to $p^2 - 1$ being divisible by a relatively large power of two. However, in these cases, most of the primes dividing $p^2 - 1$ are relatively large and would therefore give rise to slower isogeny computations during the SQISign protocol [13].

The 254-bit prime $p = 2r^3 - 1$ with $r = 20461449125500374748856320$ shown below is the best prime we found for NIST Level I. All the specific criteria that we need for a SQISign parameter set are met, while obtaining an attractively small signing cost metric $\sqrt{B}/f$. For this prime, we have

$$\begin{aligned} p+1 &= 2^{46} \cdot 5^3 \cdot 13^3 \cdot 31^3 \cdot 73^3 \cdot 83^3 \cdot 103^3 \cdot 107^3 \cdot 137^3 \cdot 239^3 \cdot 271^3 \cdot 523^3, \text{ and} \\ p-1 &= 2 \cdot 3^3 \cdot 7 \cdot 11^2 \cdot 17^2 \cdot 19 \cdot 101 \cdot 127 \cdot 149 \cdot 157 \cdot 167 \cdot 173 \cdot 199 \cdot 229 \cdot 337 \\ &\quad \cdot 457 \cdot 479 \cdot 141067 \cdot 3428098456843 \cdot 4840475945318614791658621. \end{aligned}$$

While the associated cofactor T here exceeds $p^{5/4}$, it does not exceed it by much. As we mentioned earlier, it might therefore be prone to signing failures and hence might not currently be suitable for SQISign. The next 255-bit prime of mention, $p = 2r^3 - 1$ with $r = 26606682403634464748953600$, is very similar to the previous prime, however the cofactor T exceeds $p^{5/4}$ by a larger margin, so would be less prone to these failures. In this case we have

$$\begin{aligned} p+1 &= 2^{40} \cdot 5^6 \cdot 11^3 \cdot 47^3 \cdot 67^6 \cdot 101^3 \cdot 113^3 \cdot 137^3 \cdot 277^3 \cdot 307^3 \cdot 421^3, \text{ and} \\ p-1 &= 2 \cdot 3^2 \cdot 19^3 \cdot 37 \cdot 59 \cdot 61 \cdot 97 \cdot 181^2 \cdot 197 \cdot 223 \cdot 271 \cdot 281 \cdot 311 \cdot 397 \cdot 547 \\ &\quad \cdot 1015234718965008560203 \cdot 3143438922304814418457. \end{aligned}$$

NIST-III parameters. We targeted 384-bit primes using $n = 3, 4$ and 6. The challenge in all three of these scenarios is finding enough twins whose product is divisible by a large power of two. With the limited yield of 128-bit twins, finding such primes is not straightforward; the example with $n = 3$ in Table 4 is the only such instance that we managed to find. The picture is somewhat similar with the 96-bit twins: while we have more of them, the success probabilities in Table 3 suggest that we need a lot more twins with a large power of two in order to produce any SQISign-friendly instances. One exceptional prime that was found in this search was the following 375-bit prime, $p = 2r^4 - 1$ with $r = 1232621228336746350727292 5184$. Here we have

$$\begin{aligned} p+1 &= 2^{77} \cdot 11^4 \cdot 29^4 \cdot 59^4 \cdot 67^4 \cdot 149^4 \cdot 331^4 \cdot 443^4 \cdot 593^4 \cdot 1091^4 \cdot 1319^4, \text{ and} \\ p-1 &= 2 \cdot 3 \cdot 5 \cdot 13 \cdot 17 \cdot 31 \cdot 37 \cdot 53 \cdot 83 \cdot 109 \cdot 131 \cdot 241 \cdot 269 \cdot 277 \cdot 283 \cdot 353 \cdot 419 \\ &\quad \cdot 499 \cdot 661 \cdot 877 \cdot 1877 \cdot 3709 \cdot 9613 \cdot 44017 \cdot 55967 \cdot 522673 \cdot 3881351 \\ &\quad \cdot 4772069 \cdot 13468517 \cdot 689025829 \cdot 30011417945673766253. \end{aligned}$$

Of the NIST Level III primes listed in Table 4, the prime that shows the most promise is the 382-bit prime $p = 2r^6 - 1$ with $r = 1189664338866214 5024$. Not only is the power of two particularly large but also the smoothness bound of the cofactor T is quite small, reflected in its small signing cost metric (when compared to other p where p^2-1 is divisible by a large power of 2). We have the factorisations

$$\begin{aligned} p+1 &= 2^{79} \cdot 3^{6} \cdot 23^{12} \cdot 107^{6} \cdot 127^{6} \cdot 307^{6} \cdot 401^{6} \cdot 547^{6}, \text{ and} \\ p-1 &= 2 \cdot 5^{2} \cdot 7 \cdot 11 \cdot 17 \cdot 19 \cdot 47 \cdot 71 \cdot 79 \cdot 109 \cdot 149 \cdot 229 \cdot 269 \cdot 283 \cdot 349 \cdot 449 \\ &\quad \cdot 463 \cdot 1019 \cdot 1033 \cdot 1657 \cdot 2179 \cdot 2293 \cdot 4099 \cdot 5119 \cdot 10243 \cdot 381343 \\ &\quad \cdot 19115518067 \cdot 740881808972441233 \cdot 832321437914821351639 21. \end{aligned}$$

NIST-V parameters. We targeted 512-bit primes using $n=4$ and 6. Once again, combining our CHM runs with $n=6$ proved to be the best option for finding SQISign parameters at this level. From the set of 85-bit twins found in the $B=547$ CHM run, the 510-bit prime $p=2r^6-1$ with $r=31929740427944870006521856$ is particularly attractive. The power of two here is the largest found from this run. Here we have

$$\begin{aligned} p+1 &= 2^{91} \cdot 19^{6} \cdot 61^{6} \cdot 89^{6} \cdot 101^{6} \cdot 139^{6} \cdot 179^{6} \cdot 223^{6} \cdot 239^{6} \cdot 251^{6} \cdot 281^{6}, \text{ and} \\ p-1 &= 2 \cdot 3^{2} \cdot 5 \cdot 7 \cdot 13 \cdot 23 \cdot 29 \cdot 31 \cdot 41 \cdot 53 \cdot 109 \cdot 149 \cdot 157 \cdot 181 \cdot 269 \cdot 317 \cdot 331 \\ &\quad \cdot 463 \cdot 557 \cdot 727 \cdot 10639 \cdot 31123 \cdot 78583 \cdot 399739 \cdot 545371 \cdot 550657 \cdot 4291141 \\ &\quad \cdot 32208313 \cdot 47148917 \cdot 69050951 \cdot 39618707467 \cdot 220678058317 \\ &\quad \cdot 107810984992771213 \cdot 177993780932160825 7. \end{aligned}$$

The 85-bit twins found in the CHM run with $B=2^{11}$ were used to try and find NIST-V parameters. The largest power of two that was found in this run which is suitable for SQISign was $f=109$. The prime with smallest signing cost metric while having a relatively large power of two is the following 508-bit prime, $p=2r^6-1$ where $r=26697973900446483680608256$. Here we have

$$\begin{aligned} p+1 &= 2^{85} \cdot 17^{12} \cdot 37^{6} \cdot 59^{6} \cdot 97^{6} \cdot 233^{6} \cdot 311^{12} \cdot 911^{6} \cdot 1297^{6}, \text{ and} \\ p-1 &= 2 \cdot 3^{2} \cdot 5 \cdot 7 \cdot 11^{2} \cdot 23^{2} \cdot 29 \cdot 127 \cdot 163 \cdot 173 \cdot 191 \cdot 193 \cdot 211 \cdot 277 \cdot 347 \cdot 617 \\ &\quad \cdot 661 \cdot 761 \cdot 1039 \cdot 4637 \cdot 5821 \cdot 15649 \cdot 19139 \cdot 143443 \cdot 150151 \cdot 3813769 \\ &\quad \cdot 358244059 \cdot 992456937347 \cdot 35324048178196536982389750 7 \\ &\quad \cdot 86010200695145744013716588914030 21. \end{aligned}$$

Comparison to prior work. The state-of-the-art implementation of SQISign uses a 254-bit prime that was found using the extended Euclidean algorithm (XGCD) [8,12] (see §2.1). With this method, it is possible to, for example, force $p \pm 1$ and $p \mp 1$ to be divisible by a large power of 2 and 3 (respectively). Indeed, with this approach, a smooth factor of size $\approx \sqrt{p}$ comes for free. Concretely, the prime p used in [13] has

$$\begin{aligned} p+1 &= 2^{65} \cdot 5^{2} \cdot 7 \cdot 11 \cdot 19 \cdot 29^{2} \cdot 37^{2} \cdot 47 \cdot 197 \cdot 263 \cdot 281 \cdot 461 \cdot 521 \cdot 3923 \cdot 62731 \\ &\quad \cdot 96362257 \cdot 3924006112952623, \text{ and} \\ p-1 &= 2 \cdot 3^{65} \cdot 13 \cdot 17 \cdot 43 \cdot 79 \cdot 157 \cdot 239 \cdot 271 \cdot 283 \cdot 307 \cdot 563 \cdot 599 \cdot 607 \cdot 619 \\ &\quad \cdot 743 \cdot 827 \cdot 941 \cdot 2357 \cdot 10069. \end{aligned}$$

This prime features a larger power of two compared to the primes that are mentioned in Table 4 and target NIST-I security. On the other hand, the cofactor T has a larger smoothness bound, $B=3923$, and a larger signing cost metric, at 0.96. Given that these two latter quantities are larger than in the primes we give, one could be tempted to think that the signing algorithm would perform better when instantiated with our primes. However, the implementation in [13], using the state-of-the-art prime above, benefits from the large power of three and the fact that

NIST security level	n	r	$\lceil\log_2(p)\rceil$	f	B	$\sqrt{B}/f$	$\log_p(T)$
NIST-I	2	1211460311716772790566574529001291776	241	49	1091	0.67	1.28
		2091023014142971802357816084152713216	243	49	887	0.61	1.28
	3	1022731837578822719958937б	251	31	383	0.63	1.31
		21611736033260878876800000	254	31	421	0.66	1.28
		20461449125500374748856320	254	46	523	0.50	1.26
		26606682403634464748953600	255	40	547	0.58	1.28
	4	34848218231355211776*	261	77	2311	0.62	1.30
NIST-III	3	1374002035005713149550405343373848576	362	37	1277	0.97	1.25
	4	5139734876262390964070873088	370	45	11789	2.41	1.26
		12326212283367463507272925184	375	77	55967	3.07	1.31
		18080754980295452456023326720	377	61	95569	5.07	1.26
		27464400309146790228660255744	379	41	13127	2.79	1.29
	6	2628583629218279424	369	73	13219	1.58	1.27
		5417690118774595584	375	79	58153	3.05	1.27
		11896643388662145024	382	79	10243	1.28	1.30
	12	2446635904*	376	85	9187	1.13	1.29
NIST-V	4	114216781548581709439512875801279791104*	507	65	75941	4.24	1.26
		123794274387474298912742543819242587136*	508	41	15263	3.01	1.29
	6	9469787780580604464332800	499	109	703981	7.70	1.25
		12233468605740686007808000	502	73	376963	8.41	1.28
		26697973900446483680608256	508	85	150151	4.56	1.26
		31929740427944870006521856	510	91	550657	8.15	1.25
		41340248200900819056793600	512	67	224911	7.08	1.28
	12	5594556480768*	510	97	88469	3.07	1.29
	18	335835120*	511	73	24229	2.13	1.29

Table 4: A table of SQISign parameters $p = p_n(r)$ found using twin-smooth integers $(r, r \pm 1)$ at each security level. The f is the power of two dividing $(p^2 - 1)/2$ and B is the smoothness bound of the odd cofactor $T \approx p^{5/4}$. The r marked with an asterisk correspond to primes p not found using the CHM machinery.

most of the smooth factors are relatively small. It is therefore unclear whether our NIST-I primes would outperform the prime above. Evaluating this in detail would require an implementation comparison, see Remark 7.

As mentioned earlier, our work showcases the first credible primes for SQISign at the NIST-III and NIST-V security level. A beneficial feature about most of the primes found in Table 4 is that the majority of the smooth factors are relatively small (e.g. $B < 2^{10}$). In comparison, we expect the XGCD method to scale worse for larger security levels, i.e., requiring much larger smoothness bounds. This is similar to the analysis in [9], which shows that while the XGCD approach has reasonable smoothness probabilities for NIST-I parameters, other methods become superior for larger sizes.

Remark 7. We would ideally implement the primes using the SQISign code provided in [13] to determine how well these primes perform in practice. However, the current implementation uses specific optimisations that only apply to the particular prime that is being used. Changing the prime would require reshaping the specific optimisations in the setting of our primes. As future work, we hope to implement the primes from Table 4 and benchmark their performance in practice (particularly, when compared to the current prime used in the SQISign implementation).

Remark 8. The requirement we impose on $p^2 - 1$ being divisible by $2^f \cdot T$ is to ensure that it fits within the current implementation of SQISign. At present, the SQISign implementation has a fine-grained optimisation of their generalised KLPT algorithm tied to the setting with $\ell = 2$. In general, one could allow $p^2 - 1$ to be instead divisible by $\ell^f \cdot T$, for a small prime ℓ. This could open new avenues to find SQISign-friendly primes, but would require a reconfiguration of the SQISign code. In addition, as noted in the original KLPT article [15, §4.7], this imposition could be overly restrictive: if we instead have a decomposition of the form $L \cdot T$ for a smooth L with $\gcd(L, T) = 1$, then one could still implement SQISign in theory. For example, using the prime found with $r = 20910230141429718023578160841527132 16$ from Table 4, we could use $L = 2^{49} \cdot 3^4 \cdot 5 \mid p^2 - 1$ and still have a large enough smooth factor T to exceed $p^{5/4}$.

6.3 Other Techniques for Finding SQISign Parameters

As seen in §2.1, we can collect twin smooth integers via different methods, and use them as inputs to $p_n(x)$ to search for primes. Though these alternative methods are expected to have greater smoothness bound, they have certain advantages. Namely, we are able to force larger powers of 2 into $p^2 - 1$ and search for twin smooths of large bitsizes (targeting NIST-III and -IV).

Although we expect most primes in this section to perform worse when instantiated in SQISign compared to the primes from §6.2, their concrete performance cannot be evaluated with the software from [12,13] (see Remark 7). In this section, we present the best primes found with each approach in the hopes that future implementations of SQISign benefit from a larger pool of potential primes to choose from.

XGCD twin smooths. For generating smaller twins, the XGCD approach can be used to yield relatively high smoothness probabilities. Although this increases the smoothness bound compared to CHM, we can choose smooth factors of roughly n bits combined when searching for n-bit twin smooths. This allows us to force larger powers of 2.

As an example, the 261-bit prime $p = 2r^4 - 1$ with $r = 34848218231355211776$ was found using this approach. Here we have

$$\begin{aligned} p + 1 &= 2^{77} \cdot 3^{20} \cdot 23^4 \cdot 151^4 \cdot 157^4 \cdot 233^4 \cdot 2153^4, \text{ and} \\ p - 1 &= 2 \cdot 5^2 \cdot 17 \cdot 41 \cdot 61 \cdot 71 \cdot 97 \cdot 101^2 \cdot 113 \cdot 137 \cdot 257 \cdot 263 \cdot 313 \cdot 353 \cdot 547 \cdot 853 \\ &\quad \cdot 1549 \cdot 2017 \cdot 2081 \cdot 2311 \cdot 3019 \cdot 24989 \cdot 58601 \cdot 5511340166779281313. \end{aligned}$$

This prime is similar to the primes found in [13], giving a smaller smoothness bound and a larger power of 2 compared to the state-of-the-art. However, it exceeds the size of 256 bits, and thus we expect it to perform significantly worse due to the fact that representations of values in $\mathbb{F}_p$ require an additional register in this case. Additionally, a large majority of the factors in $p^2 - 1$ are relatively large, making isogeny computations rather slow. This is consistent with the primes in [13].

PTE twin smooths. As the number of 128-bit twins that were found using CHM is relatively small, in some cases we were not able to find suitable SQISign parameters. This mainly concerns the setting using $n = 4$ and finding NIST-V parameters, for which data from the CHM run with $B < 1300$ did not yield any NIST-V SQISign-friendly instances.

To find more large twins, we can use the PTE approach [9] (see §2.1) to find 2^{14}-smooth 128-bit twins, sacrificing the smaller smoothness bounds that were used during our CHM runs. In total, we found 3,648 such 128-bit twins that resulted in a prime of the form $p = 2r^4 - 1$. Of these, two primes show strong potential to be used in SQISign and are thus also given in Table 4.

Larger values of n. We could also consider finding primes of the form $p = 2r^n - 1$ for larger values of n, where the only restriction is that r is a smooth number. Compared to the previous ideas this restriction decreases the amount of guaranteed smoothness, but if n is chosen carefully then we can obtain increased smoothness probabilities. The polynomial $p_n(x)^2 - 1$ is highly related to the cyclotomic polynomials Φ_d for $d \mid n$ as

$$p_n(x)^2 - 1 = 2x^n(2x^n - 2) = 4x^n(x^n - 1) = 4x^n \prod_{d|n} \Phi_d.$$

Recall that Φ_d is an irreducible polynomial of degree $\varphi(d)$, where φ denotes Euler's totient function. Therefore, the largest irreducible factor of $p_n(x)^2 - 1$ is of degree $\varphi(n)$. This in turn means that the largest factor that $p = 2r^n - 1$ can possibly have is around the size of $r^{\varphi(n)} \approx p^{\varphi(n)/n}$. Therefore, we would like to minimize the value $\varphi(n)/n$.

As we allow n to increase, this value can get arbitrarily low. Indeed, setting $n = P_k$, where P_k denotes the k-th primorial, we find that

$$\frac{\varphi(P_k)}{P_k} = \prod_{i=1}^{k} \frac{p_i - 1}{p_i} = \prod_{i=1}^{k} \left(1 - \frac{1}{p_i}\right),$$

and as k goes towards infinity, we see that

$$\lim_{k\to\infty} \frac{\varphi(P_k)}{P_k} = \prod_{i=1}^{\infty} \left(1 - \frac{1}{p_i}\right) = \frac{1}{\zeta(1)},$$

where $\zeta(s)$ denotes the Riemann-zeta function, which has a pole at $s = 1$.

However, we cannot allow n to become too large; we still need a sufficiently large range of inputs, so that there exists a smooth r such that $2r^n - 1$ is prime. Therefore, consider a bound $n < B_n$ where B_n is chosen such that we can still have a large search space. Based on the multiplicative property of the totient function, the fact that $\varphi(q) = q - 1$ when q is prime, and the fact that

$$\frac{\varphi(n)}{n} = \frac{\varphi(\mathrm{rad}(n))}{\mathrm{rad}(n)},$$

where $\mathrm{rad}(n)$ denotes the square-free part of n, the optimal choices of n are in the set

$$n \in \{2^{e_1} 3^{e_2} \dots p_k^{e_k} < B_n \mid P_k < B_n < P_{k+1}, e_i \geq 1\},$$

where P_k again denotes the k-th primorial.

As an example, we look for NIST-V parameters $p \in [2^{500}, 2^{512}]$. If we want at least a range of size 2^{25} such that $2r^n - 1 \in [2^{500}, 2^{512}]$, we see that we have to have $n < B_n = 20$. Therefore, our set of optimal choices of n becomes

$$n \in \{2 \cdot 3, 2^2 \cdot 3, 2 \cdot 3^2\} = \{6, 12, 18\}.$$

Using $n = 6$, the range of suitable r-values becomes large enough that we cannot search through all of them. Thus, searches would require further restrictions on the suitable r-values, such as only considering twin-smooths.

For $n \in \{12, 18\}$, we can exhaust the full search space, and obtain several promising candidates. These are include in Table 4. Among all of these, the 510-bit prime $p = 2r^{12} - 1$ with $r = 5594556480768$ seems very suitable for NIST-V. It has a low cost factor and has a large power of three, which could be beneficial for SQISign implementations. Here we have

$$\begin{aligned}
p + 1 &= 2^{97} \cdot 3^{60} \cdot 239^{12} \cdot 571^{12} \cdot 659^{12}, \text{ and} \\
p - 1 &= 2 \cdot 5^2 \cdot 7 \cdot 13^2 \cdot 17 \cdot 19 \cdot 43 \cdot 83 \cdot 103 \cdot 109 \cdot 139^2 \cdot 151 \cdot 223 \cdot 277 \cdot 317 \cdot 1249 \\
&\quad \cdot 1373 \cdot 2311 \cdot 3067 \cdot 4133 \cdot 28279 \cdot 28447 \cdot 40087 \cdot 60089 \cdot 69073 \cdot 88469 \\
&\quad \cdot 2226517 \cdot 5856073 \cdot 6242671 \cdot 14237127193 \cdot 25752311173 \\
&\quad \cdot 63101553683977 \cdot 3838024984443399866250384 1.
\end{aligned}$$

Remark 9. Unlike the CHM method and similar methods, we cannot generate more values to input into this technique, as the amount is small enough to quickly exhaust the full search space. This is in stark contrast to CHM, which could potentially - given more computing power - generate more twin smooths of given sizes to give new suitable SQISign parameters. Hence, we conclude that the CHM method with smaller values of n will ultimately give rise to new, better SQISign parameters than the ones found with the higher values of n.

Acknowledgements. We thank Joost Renes for several helpful discussions about the CHM algorithm, and Luca De Feo for helpful comments on SQISign prime requirements during the preparation of this work.

References

1. W. D. Banks and I. E. Shparlinski. Integers with a large smooth divisor. *arXiv preprint math/0601460*, 2006.
2. D. J. Bernstein, L. De Feo, A. Leroux, and B. Smith. Faster computation of isogenies of large prime degree. *Open Book Series*, 4(1):39–55, 2020.
3. T. Bingmann. TLX: Collection of sophisticated C++ data structures, algorithms, and miscellaneous helpers, 2018. https://panthema.net/tlx, retrieved Oct. 7, 2020.
4. AA Buchstab. Asymptotic estimates of a general number-theoretic function. *Mat. Sb*, 44(2):1239–1246, 1937.
5. W. Castryck and T. Decru. An efficient key recovery attack on SIDH (preliminary version). Cryptology ePrint Archive, Paper 2022/975, 2022. https://eprint.iacr.org/2022/975.
6. J. B. Conrey and M. A. Holmstrom. Smooth values of quadratic polynomials. *Experimental Mathematics*, 30(4):447–452, 2021.
7. J. B. Conrey, M. A. Holmstrom, and T. L. McLaughlin. Smooth neighbors. *Experimental Mathematics*, 22(2):195–202, 2013.
8. C. Costello. B-SIDH: supersingular isogeny Diffie-Hellman using twisted torsion. In *ASIACRYPT*, volume 12492, pages 440–463. Springer, 2020.
9. C. Costello, M. Meyer, and M. Naehrig. Sieving for twin smooth integers with solutions to the Prouhet-Tarry-Escott problem. In *EUROCRYPT*, pages 272–301. Springer, 2021.
10. N. G. de Bruijn. On the number of positive integers $\leq$ x and free of prime factors $> y$, ii. *Indag. Math*, 38:239–247, 1966.
11. K. Dickman. On the frequency of numbers containing prime factors of a certain relative magnitude. *Arkiv for matematik, astronomi och fysik*, 22(10):A–10, 1930.
12. L. De Feo, D. Kohel, A. Leroux, C. Petit, and B. Wesolowski. SQISign: compact post-quantum signatures from quaternions and isogenies. In *ASIACRYPT*, volume 12491, pages 64–93. Springer, 2020.
13. L. De Feo, A. Leroux, and B. Wesolowski. New algorithms for the Deuring correspondence: SQISign twice as fast. Cryptology ePrint Archive, Report 2022/234, 2022. https://ia.cr/2022/234.
14. D. Jao and L. De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *PQCrypto*, volume 7071, pages 19–34. Springer, 2011.
15. D. Kohel, K. Lauter, C. Petit, and J. Tignol. On the quaternion-isogeny path problem. *LMS Journal of Computation and Mathematics*, 17(A):418–432, 2014.
16. D. H. Lehmer. On a problem of Störmer. *Illinois Journal of Mathematics*, 8(1):57–79, 1964.
17. F. Luca and F. Najman. On the largest prime factor of x^2-1. *Mathematics of computation*, 80(273):429–435, 2011.
18. L. Maino and C. Martindale. An attack on SIDH with arbitrary starting curve. Cryptology ePrint Archive, Paper 2022/1026, 2022. https://eprint.iacr.org/2022/1026.
19. D. Robert. Breaking SIDH in polynomial time. Cryptology ePrint Archive, Paper 2022/1038, 2022. https://eprint.iacr.org/2022/1038.
20. C. Størmer. Quelques théorèmes sur l'équation de Pell $x^2 - dy^2 = \pm 1$ et leurs applications. *Christiania Videnskabens Selskabs Skrifter, Math. Nat. Kl*, (2):48, 1897.
21. G. Tenenbaum. Integers with a large friable component. *Acta arithmetica*, 124:287–291, 2006.
22. G. Tenenbaum. *Introduction to analytic and probabilistic number theory*, volume 163. American Mathematical Soc., 2015.
23. The National Institute of Standards and Technology (NIST). Submission requirements and evaluation criteria for the post-quantum cryptography standardization process, December, 2016.