Quantum Security of the Legendre PRF

Paul Frixons[1,2] and André Schrottenloher[3]

[1] Orange Labs, Caen, France
[2] Inria, Paris, France
firstname.lastname@inria.fr
[3] Cryptology Group, CWI, Amsterdam, The Netherlands
firstname.lastname@m4x.org

Abstract. In this paper, we study the security of the Legendre PRF against quantum attackers, given classical queries only, and without quantum random-access memories. We give two algorithms that recover the key of a shifted Legendre symbol with unknown shift, with a complexity smaller than the exhaustive search of the key. The first one is a quantum variant of the table-based collision algorithm. The second one is an *offline* variant of Kuperberg's abelian hidden shift algorithm. We note that the latter, although asymptotically promising, is not currently the most efficient against practical parameters.

Keywords: Legendre PRF, quantum cryptanalysis, quantum algorithms, abelian hidden shift problem, Kuperberg's algorithm.

1 Introduction

Let P be a prime number and $a \in \mathbb{F}_P$. The *Legendre symbol* of a modulo P is defined as $\left(\frac{a}{P}\right) = 1$ if a is a square modulo P and -1 otherwise (by convention, $\left(\frac{0}{P}\right) = 1$). Its use in cryptography was first proposed by Damgård in [19], who conjectured the hardness of the following problem:

Problem 1 (Legendre sequence randomness). Given a sequence of consecutive Legendre symbols starting at some given value $a \in \mathbb{F}_P$, of some length $m \in \mathsf{poly}\left(\log_2 P\right)$:

$$\left(\frac{a}{P}\right), \left(\frac{a+1}{P}\right), \ldots, \left(\frac{a+m-1}{P}\right), \text{ then find } \left(\frac{a+m}{P}\right) .$$

That is, consecutive Legendre symbols form a pseudo-random sequence of bits. A similar problem can be defined for the *Jacobi symbol*, which is a generalization of the Legendre symbol to a composite basis $N = P_1 \times \ldots \times P_r$: $\left(\frac{a}{N}\right) := \prod_i \left(\frac{a}{P_i}\right)$.

The Legendre PRF. The conjecture of Damgård naturally leads to the definition of a pseudo-random function based on the *shifted Legendre symbol*, as in [18]:

$$\begin{cases} F_{\text{Leg}} & : \mathbb{F}_P \times \mathbb{F}_P \to \{-1, 1\} \\ & (s, x) \quad \mapsto \quad \left(\frac{s+x}{P}\right) \end{cases}$$

or as in [21], by remapping $\{-1, 1\}$ on $\{0, 1\}$. For a given secret s, distinguishing $F_{\text{Leg}}(s, x)$ from random values is the *decisional shifted Legendre symbol problem* (DSLS), the decisional version of the *shifted Legendre symbol problem* (SLS), which asks for the recovery of s. At the moment, no separation between the SLS and the DSLS is known. In this chapter, as in previous works, we will focus on solving the SLS.

Problem 2 (Shifted Legendre Symbol). Let P be a prime number. Given query access to the function $F_{\text{Leg},s} : x \mapsto \left(\frac{x+s}{P}\right)$ for some secret $s \in \mathbb{F}_P$, find s.

The Legendre symbol PRF has recently regained significant interest with the proposal of Grassi *et al.* [21] to use it in a multi-party computation scenario. They showed precisely that there existed a simple MPC protocol to compute the PRF on secret-shared data, using the malleability of the Legendre symbol.

Since then, the PRF has been considered for use in the Ethereum blockchain, and the Ethereum foundation has proposed several challenges to encourage cryptanalysis research [20]. While the Ethereum challenges are not explicitly concerned with quantum adversaries, this is the case of the LegRoast signature scheme proposed in [6], which adapts the Picnic construction [15] with the Legendre PRF. Solving the SLS problem allows to break these different schemes; however, in both cases the adversary cannot make chosen-plaintext queries to the PRF. In the Ethereum challenges he only knows a (large) sequence of successive Legendre symbols, a condition that can be satisfied by all the algorithms studied in this chapter. In LegRoast, he only learns the evaluation of the PRF on a few random, uncontrolled inputs.

Here, we focus on the SLS problem, and so, the security of the Legendre PRF, against a quantum adversary. The algorithms considered derive their advantage from making a large number of queries, and so, they do not impact all applications of the PRF in the same way (e.g., LegRoast does not seem to be affected).

Previous Results. The SLS problem is a particular case of the *shifted character problem*, where the Legendre symbol is replaced by any multiplicative character of a finite field. This problem was studied by van Dam *et al.* [18] in the quantum setting. When *superposition query access* to the shifted character is given, they showed that the problem could be solved in polynomial time, using a single query.

However, when only *classical* queries are available, the SLS problem is believed to remain intractable for a quantum attacker, and was conjectured so by van Dam *et al.* and by Grassi *et al.*.

In the classical setting, several authors have studied and improved the Legendre PRF key-recovery attacks [27,25,5]. We give a summary in Table 1, and recapitulate the detailed complexities in Table 2 in Appendix. The most advanced

results were obtained by Beullens *et al.* [5] and concurrently by Kaluderović *et al.* [25,26]. Given a sequence of M Legendre symbols, they recover the secret in about $\widetilde{\mathcal{O}}\left(\frac{P}{M^2}\right)$ operations. Their technique is a *table-based collision search*, whose principle will be reviewed in Section 3. Generalizations of the Legendre PRF (e.g. when $x+s$ is replaced by a polynomial in x) have also been studied.

Even more recently, Seres, Horváth and Burcsi [35] showed that the problem of recovering the key of a Legendre PRF is equivalent to solving some multivariate quadratic system of equations. So far this approach did not yield better attacks than those of [25,5], as dedicated algebraic attacks seem inefficient against this system.

On the use of QRACM. In the quantum setting, it was proposed [25] to use a quantum version of the table-based collision search (some remarks were also made in [5]). However, in the same way that the classical attack uses a large table, the quantum attack will use a table of similar size. This table requires the model of *classical memory with quantum random-access* (QRACM). Although this is a powerful memory model, it is required by many quantum algorithms, e.g., for BHT quantum collision search [12], which makes it theoretically worth studying.

No practical, scalable implementation of QRACM exists at the moment, and near-term quantum architectures are expected to consist only of error-corrected quantum circuits of small width. Several authors envision an advantage coming from parallel circuits rather than memory usage [3,24]. In this context, QRACM can be seen as a conservative assumption, that cannot be always accurate. This is why, more recently, quantum cryptanalytic algorithms have been developed that aim for an advantage over classical algorithms, while using standard computing qubits only.

To our knowledge, there are two main examples of this. The first one is the technique applied in [14] to collision search and [23] for the subset-sum problem. In this setting, QRACM is replaced by a large classical memory, which is accessed sequentially. While QRACM essentially accesses a memory of size M in time $\mathsf{polylog}(M)$, these algorithms access a memory of size M in time $\widetilde{\mathcal{O}}(M)$. This is the technique that we apply in Section 3.

The second one, much more dependent on the structure of the problem, uses a hidden shift property, as in [8]. In this setting, there is no large-scale memory anymore, neither classical nor quantum. Thus having QRACM or not becomes less of a concern. This is the technique that we apply in Section 5. Although the complexity scales differently with the number of queries, the algorithm eventually reaches the same limit as the QRACM-heavy table-based collision search when M increases to $P^{1/3}$. However, it suffers from a subexponential factor in time complexity. But this factor might become competitive compared to the cost of maintaining and accessing a large QRACM.

Table 1. Lowest time complexity (in P) achieved by the known SLS algorithms and corresponding memory and queries. For ease of comparison, we dismiss polynomial factors in $\log_2 P$.

Method	Queries	Time	Memory	Source
		Classical algorithms		
Pollard's rho	$\sqrt{P}$	$\sqrt{P}$	P	[27]
Table	$P^{1/4}$	$\sqrt{P}$	$\sqrt{P}$	[5,25]
		Quantum algorithms		
Sup. queries	2	P	P	[18]
Table (QRACM)	$P^{1/6}$	$P^{1/3}$	$P^{1/3}$ QRACM	[25]
Grover search	P	$\sqrt{P}$	P qubits	Sec. 3.2
Table (no QRACM)	$P^{3/14}$	$P^{3/7}$	$P^{3/14}$ classical + P qubits	Sec. 3.3
Offline-DAHS	$P^{1/3}$	$(2^{\frac{4}{3}\sqrt{2\log_2 3}\sqrt{\log_2 P^{1/3}}})P^{1/3}$	$2^{\sqrt{2\log_2 3}\sqrt{\log_2 P^{1/3}}}$ qubits	Sec. 5.3

2 Preliminaries of Quantum Computing

2.1 Quantum Search with an Approximate Test

Next, we use another result on Grover search using an *approximate test oracle*. This is an important object in the context of *offline* search algorithms such as offline-Simon [8] or offline-DAHS below.

We consider a boolean function $f : \{0,1\}^n \to \{0,1\}$ and an oracle O_f that uses an *ancillary state* $|\psi\rangle$. This ancillary state is required by the oracle to perform its computations, and it must be preserved. So one would expect to map: $|x\rangle|\psi\rangle \xmapsto{O_f} (-1)^{f(x)}|x\rangle|\psi\rangle$. However, the exact O_f cannot be implemented, only an approximation O'_f.

Definition 1. *An* approximate test oracle *for f, denoted O'_f, is a unitary operator that maps:*

$$\forall x \in \{0,1\}^n, |x\rangle|\psi\rangle \xmapsto{O'_f} (-1)^{f(x)}|x\rangle|\psi\rangle + |\delta_x\rangle$$

where $|\psi\rangle$ is the predefined ancillary state *which must be preserved by the oracle, and* $\max_x \| |\delta_x\rangle \| \leq \epsilon$.

That is, on each input vector, the oracle incurs a *uniform* error of amplitude ϵ (the same for all basis states). If this error is small enough, an Amplitude Amplification using O'_f instead of O_f will not "see" the difference. In our case, contrary to [8], we will also start from an *approximate initial state*. We summarize these needs with the following.

Theorem 1 (Quantum search with approximate test). *Let* $\mathcal{A}, a, t = \left\lfloor \frac{\pi}{4\theta_a} \right\rfloor$ *be defined as in Theorem* **??**. *Let* f *be a boolean function that tests if an output of* $\mathcal{A}$ *is "good", and let* O'_f *be an approximate oracle for* f *with error* ϵ*, using an ancillary state* $|\psi\rangle$*, as per Definition 1. Let* $|\psi'\rangle$ *be a quantum state such that* $\| \, |\psi\rangle - |\psi'\rangle \, \| \leq \nu$*. We run the following algorithm:*

$$\left((\mathsf{I} \otimes \mathcal{A})(\mathsf{I} \otimes O_0)(\mathsf{I} \otimes \mathcal{A}^\dagger)O'_f\right)^t |\psi'\rangle \quad ,$$

that is, t *iterations of "approximate" quantum search using* O'_f*, where* I *is the identity operator applied to the ancillary state* $|\psi\rangle$*. Then measuring the output yields a good result with probability greater than* $(1 - t\epsilon - \nu)^2 \max(1 - a, a)$.

For completeness, we give a proof of Theorem 1 in Appendix 8. It consists in bounding the euclidean distance, step by step, between the current state of the "ideal" version of the algorithm (starting from $|\psi\rangle$ and applying O_f) and the "real" one (starting from $|\psi'\rangle$ and applying O'_f).

Corollary 1. *If* $\nu \leq \frac{1}{4}$ *and* $\epsilon \leq \frac{1}{4t}$*, then the procedure of Theorem 1 succeeds with probability* $\geq \frac{1}{8}$*.*

Proof. We use simply that $\sin^2((2t+1)\theta) \geq \frac{1}{2}$ and $1 - t\max(\epsilon_{FN}, \epsilon_{FP}) - \epsilon \geq 1 - \frac{1}{4} - \frac{1}{4} = \frac{1}{2}$. □

We stress that Theorem 1 and Corollary 1 contain two important features, that are both crucial for solving the SLS problem: • the test oracle can be imperfect, as soon as the error is *uniform* and *exponentially small* (when t is exponential); • the ancillary state used by the approximate test can also be approximated, and the corresponding error ν *can be a constant.*

3 Table-based Attacks

In this section, we first recall the classical *table-based collision search* of [5,25] (Section 3.1), and the idea of *early abort* in a Grover search (Section 3.2), which is a folklore technique that we will use to save logarithmic factors. Next, we introduce our new *quantum table-based collision search*. It combines the ideas of the classical attack and the quantum collision search algorithm of [14]. The notations follow those of Beullens *et al.* [5].

As all the attacks studied in this paper, the table-based collision attacks use chosen, but non-adaptive queries to the Legendre PRF. More precisely, we assume that we can query $M < \sqrt{P}$ consecutive values of the Legendre PRF, of the form $\left(\frac{x+s}{P}\right)$ where $0 \leq x \leq M-1$ and s is the secret. This is actually the setting of the Ethereum challenges [20].

3.1 Classical Algorithm

We set $m = 3\lceil \log_2 P \rceil$. The attack parses the M sequential Legendre symbols to create *L-sequences*: words of m bits that allow to discriminate a guess of s. The definition of L-sequences is from [5] and we slightly simplify it. We define:

$$L_a = \left(\left(\frac{a}{P}\right), \left(\frac{a+1}{P}\right), \ldots, \left(\frac{a+m-1}{P}\right) \right) \ .$$

Assuming that the Legendre PRF is "sufficiently random", and that m is big enough, a collision of L-sequences implies the equality of their parameters: $L_a = L_b \implies a = b$. With our choice of m, we will assume that *no random collisions occur at all*. This is a stronger heuristic than the one commonly used in the classical cryptanalysis of the Legendre PRF (see [5], Assumption 1). It will simplify our analysis of quantum algorithms.

Heuristic 1 *For all $a, b \in \mathbb{Z}_P^*$, $L_a = L_b \implies a = b$.*

In order to recover the secret s, one then looks for a collision between an L-sequence of unknown parameter (depending on s) and an L-sequence of known parameter. A basic attack could use the M queries to obtain $M - m$ L-sequences, and then, evaluate random L-sequences in search for a collision. This would give a complexity $\mathcal{O}\left(M + \frac{P}{M}\right)$ (as two L-sequences can only collide with probability $\frac{1}{P}$).

The attacks of [5,25,26] improve the trade-off between M and the time complexity, by extracting more L-sequences from the available queries. This uses the multiplicativity of the Legendre symbol. The attacks run as follows:

1. From the M consecutive Legendre symbols, extract $\frac{M^2}{m}$ L-sequences of the form:

$$\begin{aligned} &\left(\left(\frac{a+s}{P}\right), \left(\frac{a+b+s}{P}\right), \ldots, \left(\frac{a+b(m-1)+s}{P}\right) \right) \\ &= \left(\frac{b}{P}\right) \left(\left(\frac{a/b+s/b}{P}\right), \left(\frac{a/b+1+s/b}{P}\right), \ldots, \left(\frac{a/b+(m-1)+s/b}{P}\right) \right) \\ &= \left(\frac{b}{P}\right) L_{(s/b+a/b) \bmod P} \ . \end{aligned}$$

   Since we can use $b \leq \left\lfloor \frac{M}{m} \right\rfloor$ and $a < M - bm + 1$, the number of sequences is increased quadratically with respect to the naive extraction.
2. Store all the extracted sequences $(L_{(s/b+a/b) \bmod P}, a, b)$ in a table.
3. Sample c at random until (L_c, a, b) appears in the table for some a, b. Such a collision yields a candidate key s such that: $s/b + a/b = c \implies s = cb - a \bmod P$. We can then test if this candidate is the good one with a few more computations. With Heuristic 1, there are no false positives, and s is the right key.

This classical attack requires M^2 storage for the table, and expectedly mP/M^2 samples must be tested in Step 3 before a collision occurs. Thus Step 3 requires $\mathcal{O}\left(m^2P/M^2\right)$ Legendre symbol computations. Further optimizations allow to reduce the memory to M^2/m and to amortize the cost of computing Legendre symbols in an iteration of the loop.

Quantum Version with QRACM. This procedure yields a quantum attack as proposed in [25]. The precomputation stage (Step 1) is unchanged, but now Step 3 is a quantum search. Instead of running $\mathcal{O}\left(mP/M^2\right)$ classical iterations, we need only $\mathcal{O}(\sqrt{mP/M^2})$ iterates, each of which performs $\mathcal{O}\left(m\right)$ Legendre symbol computations and a memory access. This can only be efficient if we use classical memory with quantum random-access (QRACM).

3.2 Quantum Search with Early Abort

Using the same M classical sequential queries, but without QRACM, the first quantum attack available is a direct quantum search of the secret s. We query the L-sequence L_s (thus using only m data) and search for $x \in \mathbb{Z}_p^*$ such that $L_x = L_s$. We simply apply Theorem **??**, where the amplified algorithm $S_{\mathbb{Z}_P^*}$ consists in sampling an element $x \in \mathbb{Z}_P^*$ at random: $S_{\mathbb{Z}_P^*}\left|0\right\rangle = \sum_{x\in\mathbb{Z}_P^*}\left|x\right\rangle$, and the test f consists in checking if $L_x = L_s$. By Heuristic 1, there is only one such solution; thus the algorithm requires $\mathcal{O}(m\sqrt{P})$ (quantum) Legendre symbol computations, in total $\mathcal{O}(m^3\sqrt{P})$ quantum gates.

Early-aborting. In a classical search for a sequence matching L_s, we can stop the computation of L_x at the first bit that does not match. This reduces the average number of Legendre symbols computed from m to a constant. In the quantum setting, this folklore idea allows to amortize the factor m down to $\log_2 m$.

We select a constant $i \leq m$ and define a subset of $\mathbb{Z}_P^*$:

$$X_i = \{x \in \mathbb{Z}_P^*, L_x \text{ and } L_s \text{ match on the first } i \text{ bits }\}.$$

We know that s belongs to X_i, and by the pseudorandomness of the Legendre PRF, that X_i is roughly of size $\frac{P}{2^i}$. Furthermore, testing if a given x is in X_i requires to compute only i Legendre symbols, not m. This allows to filter out most of the incorrect values.

We first use Theorem **??** to create an "inner" search: an algorithm S_{X_i} that samples from X_i. This algorithm runs a quantum search over $x \in \mathbb{Z}_P^*$ with $\mathcal{O}(\sqrt{2^i})$ iterations, computing i Legendre symbols each. Since we do not know exactly the size of X_i, we must run the search with a fixed number of iterations, and the output state is not exactly the uniform superposition over X_i. However, we only need to sample from X_i with constant probability α. Afterwards, the output of S_{X_i} is "good" for us (equal to s) with probability $\alpha\frac{2^i}{P}$, where α is a constant.

We can thus use Theorem **??** again. We amplify S_{X_i}, using $\mathcal{O}\left(\sqrt{\frac{P}{2^i}}\right)$ iterations in total. Each iteration contains a computation of S_{X_i} and $m-i$ Legendre symbols.

This gives a total complexity:

$$\mathcal{O}\left(\sqrt{\frac{P}{2^i}}\left(\sqrt{2^i}(iL)+(m-i)L\right)\right) .$$

Taking $i = 2\log_2 m$ and simplifying gives: $\mathcal{O}\left(\log_2 m\sqrt{P}L\right)$.

3.3 Distinguished Collisions

Since we do not assume QRACM, we have to modify the table-based collision strategy if we are to beat the square-root complexity given by Grover search. We will use the strategy of [14] for multi-target preimage search, and adapt the algorithm of Section 3.1 as follows:

1. From the M Legendre symbols, extract $\frac{M^2}{m}$ L-sequences.
2. Store only the $\frac{M^2}{m2^t}$ "distinguished" sequences that start with t zeroes (an arbitrary choice).
3. Sample x such that L_x is distinguished, until it matches one of the stored sequences.

We use Amplitude Amplification again. First, we built a quantum algorithm S_D that samples x such that L_x is distinguished. We can do that in time: $\mathcal{O}(\log_2 t\sqrt{2^t}L)$ using an early-aborted quantum search. Again, S_D is not exact, and if we measure its output, we get a distinguished L_x with constant probability only. But this is enough.

We next estimate the probability that S_D returns a "good" output, that is, an x such that L_x collides with one of the stored sequences. There are on average $P/2^t$ distinguished sequences, and the probability to collide on the table is roughly $\frac{M^2/(m2^t)}{P/2^t}$. Depending on the exact number of distinguished sequences, this probability also deviates from the expectation, but not more than by a constant.

Next, to test if a distinguished L_x matches one of the stored sequences, we use a sequential circuit containing quantum gates *controlled by classical values*. A single L_a can be compared to the current L_x with a comparator using $\mathcal{O}(m)$ gates; we repeat this for all distinguished sequences of our table.

Assuming that we use all the data, this yields an algorithm of complexity:

$$\mathcal{O}\left(M^2+\sqrt{\frac{P}{2^t\times\frac{M^2}{m2^t}}}\left(\log_2 t\sqrt{2^t}L+mL+\frac{M^2}{m2^t}m\right)\right) .$$

Note that the outer number of iterations has been reduced, because two distinguished sequences have a higher probability to collide than two random sequences.

By taking $t = \frac{4}{3}\log_2(M/m)$ we get a complexity:

$$\mathcal{O}\left(M^2 + \sqrt{\frac{Pm}{M^2}}\left(\frac{M}{m}\right)^{2/3}(\log_2\log_2 M)m^2\right) = \mathcal{O}\left(M^2 + \frac{\sqrt{P}}{M^{1/3}}m^{11/6}\log_2\log_2 M\right) .$$

Note that a memory of size M is required during Step 1 (extraction), and $M^{2/3}m$ during Step 3 (search). Both are only classical. Also, the memory of Step 3 is accessed only once per iteration ($\sqrt{\frac{Pm}{M^2}}$ in total) and in a sequential way. The complexity of the classical table-based collision decreases with the available data M, from $\widetilde{\mathcal{O}}(P)$ down to $\widetilde{\mathcal{O}}\left(P^{1/2}\right)$ when $M = P^{1/4}$. This quantum variant decreases from $\widetilde{\mathcal{O}}\left(P^{1/2}\right)$ to $\widetilde{\mathcal{O}}\left(P^{3/7}\right)$, reaching this minimum when $M = P^{3/14}$ data is available.

4 A Reversible Version of Kuperberg's Algorithm

In this section, we recall the *abelian hidden shift problem* and Kuperberg's first algorithm [28]. It is usually depicted with intermediate measurements and classical postprocessing. Our goal is to make it reversible. Thus, although we ultimately obtain the same asymptotic complexity, our presentation (from Section 4.3 onwards) is new and differs significantly from the ones of [28,9].

4.1 The Abelian Hidden Shift Problem

Quantum algorithms for *hidden period* or *hidden shift* problems have seen numerous applications in quantum cryptanalysis. As an example, Shor's algorithm [36] solves the *abelian hidden period* (or abelian hidden subgroup) problem in polynomial time: given a function f with domain $(G, +)$, an abelian group, such that $f(x + s) = f(x)$ for some $s \in G$, find s. But the problem becomes harder if we look for an abelian *shift* between two functions.

Problem 3 (Abelian hidden shift). Let $(G, +)$ be an abelian group, X a set and $f, g : G \to X$ a pair of injective functions such that: $\exists s, \forall x \in G, g(x) = f(x + s)$. Then find the *shift* s.

As it was already remarked in [18], the SLS problem is an instance of Problem 3, where: $g(x) = L_{s+x}$, obtained by m queries to the PRF $F_{\mathrm{Leg},s}$, and $f(x) = L_x$, obtained by m computations of Legendre symbols modulo P. In this paper, we will not use Kuperberg's algorithm to solve directly the SLS, but a *decisional* version of Problem 3.

Problem 4 (Decisional abelian hidden shift). Let $f, g : \mathbb{Z}_{2^n} \to X$ be two injective functions such that either: $\exists s, \forall x, f(x + s) = g(x)$, or $Im(f) \cap Im(g) = \emptyset$. Decide which is the case.

4.2 Kuperberg's Algorithm made Reversible

In [28], Kuperberg designed a subexponential-time algorithm to solve Problem 3 (and so Problem 4), using quantum oracle access to f and g. The original algorithm ran in time $\widetilde{\mathcal{O}}(2^{\sqrt{(2\log_2 3)\log_2 |G|}})$. Multiple subsequent works have changed the value in the exponent and given trade-offs between classical and quantum computations [33,29,16,7,9,32,10]. If queries, classical and quantum time are counted equally, then the best complexity known to date is $\widetilde{\mathcal{O}}(2^{\sqrt{2n}})$ with Kuperberg's *collimation sieve* [29]. In this paper, we focus on the earliest algorithm for its simplicity, but we believe that further asymptotic improvements might come from the collimation sieve.

As we focus on Problem 4, we will work with an abelian group of the form $G = \mathbb{Z}_{2^n}$ for some n. We note $M = 2^n$ the cardinality of the group. Note that the generalization to an arbitrary abelian group would be technical, but without significant incidence on the time complexity [10]. The assumption that the functions are injective is also helpful for our study, but not strictly necessary.

Sample Database. We define a *sample state* as:

$$|\psi_{f,g}\rangle = \sum_{0\leq x\leq M-1} \Big(|0\rangle\,|f(x)\rangle + |1\rangle\,|g(x)\rangle\Big)\Big)\,|x\rangle \quad ,$$

omitting the common amplitude factor for ease of notation. One creates it with a single oracle query to O_f and O_g. Kuperberg's algorithm starts by producing $t = \widetilde{\mathcal{O}}(2^{\sqrt{\alpha n}})$ such states, where $\alpha = 2\log_2 3$ is obtained from the complexity analysis. We name $|\psi_{f,g}\rangle^{\otimes t}$ the *sample database* of (f, g): it contains all the information on f and g that we need to solve Problem 4.

Our goal is to process this state to decide whether f and g are shifted, *without destroying the database*. Measurements can always be removed from a quantum computation. Thus, we know that we can follow the standard operations of Kuperberg's algorithm, but without performing any measurement, and obtain a quantum circuit DAHS that yields the same result.

However, the standard procedure handles a lot of classical data, and performs non-trivial memory operations. This may increase significantly the time complexity in a fully reversible variant, especially since we do not want to rely on quantum RAM. We will show that reversibility costs only at most a polynomial factor in time. Given the *sample database* $|\psi_{f,g}\rangle^{\otimes t}$, the circuit DAHS finds whether f and g are shifted with constant probability of success. This probability can then be boosted by taking a polynomial number of copies of the circuit (or it can be estimated heuristically, as we do in Appendix 11). All in all, we prove the following theorem.

Theorem 2. *There exists a quantum circuit* DAHS *that maps:*

$$|\psi_{f,g}\rangle^{\otimes t}\,|b\rangle \mapsto |\psi_{f,g}\rangle^{\otimes t}\,|b \oplus \mathsf{DAHS}(f,g)\rangle + |\delta_{f,g}\rangle \quad ,$$

where $\mathsf{DAHS}(f, g) = 1$ *if and only if* f *and* g *are a shifted pair, and* $\| \, |\delta_{f,g}\rangle \, \|$ *is bounded by a constant for all* f, g *satisfying the conditions of Problem 4. With*

$t = \widetilde{\mathcal{O}}\left(2^{\sqrt{(2\log_2 3)n}}\right)$, *it contains* $\widetilde{\mathcal{O}}\left(2^{\sqrt{(2\log_2 3)n}}\right)$ *quantum gates and ancilla qubits. With a factor* r *in time and memory complexity, we can reduce the bound on* $\| \, |\delta_{f,g}\rangle \, \|$ *to* $\mathcal{O}\left(2^{-r/2}\right)$.

We devote the rest of this section to the details of this procedure. For now, we will assume that f and g are shifted, and we will go back to the other case at the end of this section.

4.3 Label States and Label Qubits

The first step in Kuperberg's algorithm is to transform all the independent sample states into *label states*, by measuring a value a in the second register and applying an M-dimensional QFT on the third register. In the reversible variant, we don't measure this a. We don't write it either for simplicity (after the QFT, it does not intervene further in the algorithm).

Since the functions are injective, there exists a single x_0 that maps to a through f and by assumption, $x_0 - s$ maps to a through g. We obtain:

$$\begin{aligned} |\phi_{f,g}\rangle &= \mathrm{QFT}_M(|0\rangle\,|x_0\rangle + |1\rangle\,|x_0 - s\rangle) \\ &= \sum_{0\le y\le M-1} \Big(\chi_M(x_0 y)\,|0\rangle + \chi_M((x_0-s)y)\,|1\rangle\Big)\,|y\rangle \\ &= \sum_{0\le y\le M-1} \chi_M(x_0 y)\Big(\,|0\rangle + \chi_M(-sy)\,|1\rangle\Big)\,|y\rangle \quad . \end{aligned}$$

Next, we would have measured the register $|y\rangle$ to obtain a random value y and a *label qubit*: $|\phi_y\rangle = |0\rangle + \chi_M(-sy)\,|1\rangle$ (up to a global phase factor). Again, we do not measure y. So we simply write label states as $|\phi_y\rangle\,|y\rangle$, keeping in mind that there is a superposition over y.

A label qubit contains some information about s, but it is not immediately exploitable, unless we can obtain specific values of y. For example, if $y = 2^{n-1}$, then the qubit is either $|0\rangle + |1\rangle$ or $|0\rangle - |1\rangle$ depending on the least significant bit of s. Recall that we are interested in *deciding* whether there is a shift or not (Problem 4). We can do that from many independent copies of $|\phi_{2^{n-1}}\rangle$.

Classical Combinations. The key step in the algorithm is when we create these wanted labels from the random initial ones. In the standard procedure, the values of the labels are known. From two label qubits $|\phi_{y_1}\rangle$ and $|\phi_{y_2}\rangle$, we can obtain $|\phi_{y_1 \pm y_2}\rangle$ with a CNOT and a measurement. This destroys both qubits and returns either $y_1 + y_2$, or $y_1 - y_2$, with probability $\frac{1}{2}$. The procedure then consists in combining pairs of labels (y_1, y_2) that maximize the expected valuation of $y_1 \pm y_2$ modulo 2. We let $\mathrm{val}_2(z) = \max\{i, 2^i | z\}$ denote this valuation.

Complexity. The complexity analysis in [28] gives that $\widetilde{\mathcal{O}}\big(2^{\sqrt{(2\log_2 3)n}}\big)$ initial label qubits are enough. Simulations in [9] showed that $2^{\sqrt{(2\log_2 3)n}}$ were essentially

enough to solve Problem 3 with constant probability, for groups of the form $\mathbb{Z}_{2^n}$. Interestingly, the algorithm is *pseudoclassical*: the combination step can be easily simulated by sampling labels at random. This allows to compute precisely how many labels will be needed for a given instance of the problem, and to obtain the corresponding success probability.

4.4 Combining two Labels Reversibly

When combining two labels, we start from the joint state:

$$|\phi_{y_1}\rangle\,|\phi_{y_2}\rangle\,|y_1\rangle\,|y_2\rangle = \Big(\,|0\rangle + \chi_M(-y_1 s)\,|1\rangle\,\Big)\Big(\,|0\rangle + \chi_M(-y_2 s)\,|1\rangle\,\Big)\,|y_1\rangle\,|y_2\rangle = \Big(\,|00\rangle + \chi_M(-y_1 s)\,|10\rangle + \chi_M(-y_2 s)\,|01\rangle + \chi_M(-(y_1+y_2)s)\,|11\rangle\,\Big)\,|y_1 y_2\rangle \quad ,$$

and we CNOT the first qubit into the second one, mapping $|10\rangle$ to $|11\rangle$ and $|11\rangle$ to $|10\rangle$. We obtain:

$$\Big(\,|0\rangle + \chi_M(-(y_1+y_2)s)\,|1\rangle\,\Big)\,|0\rangle\,|y_1\rangle\,|y_2\rangle + \chi_M(-y_2 s)\Big(\,|0\rangle + \chi_M(-(y_1-y_2)s)\,|1\rangle\,\Big)\,|1\rangle\,|y_1\rangle\,|y_2\rangle \quad .$$

In order to mimic the classical recomputation of labels, we perform a controlled addition or subtraction of y_2 in place, on the register that contains y_1. We obtain the state:

$$\Big(\,|\phi_{y_1+y_2}\rangle\,|0\rangle\,|y_1+y_2\rangle + \chi_M(-(y_1-y_2)s)\,|\phi_{y_1-y_2}\rangle\,|1\rangle\,|y_1-y_2\rangle\,\Big)\,|y_2\rangle \quad .$$

Once this operation has been performed, we get a qubit $|0\rangle$ or $|1\rangle$ that used to indicate whether we obtained the sum $y_1 + y_2$ or the difference $y_1 - y_2$: here, it is just kept along for reversibility. The register that contains y_2 has become entangled with the other and cannot be used for further combinations.

In our DAHS circuit, we will define a "combination circuit" Comb_v (Figure 1 in Appendix). We flag all the label states with additional qubits that inform us whether the label can be used for further combination or not. The circuit Comb_v then first tests that the two labels y_1, y_2 have valuation v and that their flags b_1, b_2 are equal to 1 (Flag). If this is true, then it performs an addition or subtraction in place ($\mathsf{Sub}, \mathsf{Add}$), then NOTs the flag qubit b_2 (X), since y_2 cannot be used for combination anymore.

Choosing which Labels to Combine. Among the pairs of labels y_1, y_2 having the same valuation modulo 2, we want to select those which maximize the expected valuation of $y_1 \pm y_2$. We check whether the second to last bit of $y/2^{\mathrm{val}_2(y)}$ is 0 or 1. If it is 0, then our hope is to add y to another label that maximizes the overlap of least significant bits. If it is 1, then our hope is to subtract y to another label

that overlaps with $2^n - y$ on as many least significant bits as possible. Thus we can define a function F on labels[4]:

$$F(y) = \begin{cases} (1,0) \text{ if } y = 0 \text{ or } y \text{ cannot be combined anymore} \\ (-\mathrm{val}_2(y), \mathrm{rev}(2^n - y, n)) \text{ if the second to last bit of } y/(2^{\mathrm{val}_2(y)}) \text{ is 1} \\ (-\mathrm{val}_2(y), \mathrm{rev}(y, n)) \text{ otherwise} \end{cases}$$

where $\mathrm{rev}(y, n)$ reverses the bits in y (for a total of n bits). By sorting the labels according to F, we ensure that the best pairs, such that $y_1 \pm y_2$ has the best expected valuation, are put together.

4.5 Combining All Labels

We start from a list of $t = 2^\ell$ label states for some integer ℓ. For $v = 0, \ldots, n-2$:

- We perform a reversible sorting network for F: we compute F in ancillas, perform a sorting network, and then uncompute F. We consider that the computation of F can be neglected. The sorting network is a series of comparators and swaps (controlled on the results of the comparators). The labels are moved in place. For reversibility, the outcome of each comparator must be written in a new qubit.
- we apply the combination circuit Comb_v on each pair of labels at positions $(2i, 2i+1)$ for $i \leq 2^{\ell-1}$. It writes 2ℓ new qubits that contain carries and inform whether the combinations occurred.

In practice, the combination layer at step v consumes all labels of valuation v and creates labels of higher valuation. The main difference with the classical process is that, although the labels are sorted to ensure a maximal number of zeroes after combination, since we take them 2 by 2 on arbitrary positions we might create a few suboptimal pairs. This does not change the asymptotic complexity of the procedure.

Note that all labels equal to 0 mod 2^n, and the labels that cannot be combined anymore, are moved to the bottom of the list by sorting. The labels equal to 2^{n-1} will be moved to the top. Thus, we know where to look for them.

Sorting Network. We use the *odd-even mergesort* of Batcher [2]. On input a list of 2^ℓ n-bit strings, it uses a total of $S(2^\ell) = 2^{\ell-1}\frac{\ell(\ell-1)}{2} + 2^\ell - 1 = \mathcal{O}\left(\ell^2 2^\ell\right)$ comparators and controlled SWAPs. In order to be made reversible, it also needs to write $\mathcal{O}\left(\ell^2 2^\ell\right)$ new qubits.

The Full Circuit. The full combination circuit contains $n-1$ layers of sorting, followed by layers of combination circuits: at layer i (starting from 0), we combine only the labels having valuation i. The complexity mainly depends on the sorting steps: there are in total $\mathcal{O}(n) \times \mathcal{O}(n) \times S(2^\ell) = \mathcal{O}\left(n^2 \ell^2 2^\ell\right)$ quantum gates used,

[4] Although we did not find its explicit definition in previous works, it appears in the simulation code of [9].

mainly for comparators and SWAPs. The circuit writes $nS(2^\ell)+(n-1)2^\ell$ ancillas. They are uncomputed afterwards.

After the combination, we look at the r first label registers, for some value r to choose later. We expect them to contain r copies of 2^{n-1}. As we have seen, in the shifted case, the corresponding qubits contain identical copies of $|0\rangle + |1\rangle$, or $|0\rangle - |1\rangle$ depending on the parity of s. Thus, we perform a Hadamard transform on them and test if the result is all-zero or all-one.

Note that if the first label registers do not contain the expected copies of 2^{n-1}, then *we know that the circuit has failed*. The fact that failures are detectable allows to reduce easily the probability of failure with multiple copies of the circuit: we simply take the result that hasn't failed.

4.6 Behavior in the Non-shifted Case

In the non-shifted case, by our assumptions, there is no overlap between the functions f and g. Thus, sample states are the sums of two independent parts:

$$|\psi_{f,g}\rangle = \underbrace{\Big(\sum_x |0\rangle\,|f(x)\rangle\,|x\rangle\Big)}_{|\psi_f\rangle} + \underbrace{\Big(\sum_x |1\rangle\,|g(x)\rangle\,|x\rangle\Big)}_{|\psi_g\rangle}\ ,$$

and the whole sample database $|\psi_{f,g}\rangle^{\otimes t}$ can be rewritten as a sum of the 2^t states of the form $|\psi_{h_1}\rangle \otimes \cdots \otimes |\psi_{h_t}\rangle$ where $h_t \in \{f, g\}$. By linearity, we can focus on the output of DAHS on one of these states only.

The QFT applied to $|\psi_h\rangle$ yields a state $|b\rangle \sum_y \left(\sum_x \chi_M(xy)\,|h(x)\rangle\right)|y\rangle$ where b depends only on h. Therefore, after performing the combination step, and after obtaining labels equal to 2^{n-1}, the corresponding qubits are not $|0\rangle \pm |1\rangle$, but either $|0\rangle$, or $|1\rangle$, depending on the exact sequence of combinations performed.

Recall that in the shifted case, the r qubits on which we apply the final Hadamard transform contain copies of $|0 \pm 1\rangle$. If the algorithm succeeds, these copies are obtained in all cases, so they are disentangled from the rest of the state. In contrast, in the non-shifted case, we obtain a single r-dimensional basis state depending on the sequence of combinations. Thus after applying the Hadamard transform, the total amplitude on $|0^r\rangle$ (resp. $|1^r\rangle$) is equal to $2^{r/2}$.

4.7 Bounding the Errors

The circuit DAHS is not exact. In the positive (shifted) case, it maps:

$$|\psi^t_{f,g}\rangle\,|b\rangle \xmapsto{\mathsf{DAHS}} |\psi^t_{f,g}\rangle\,|b \oplus 1\rangle + |\delta^{f,g}_{\mathrm{FN}}\rangle\,|b\rangle$$

where $\| \,|\delta^{f,g}_{\mathrm{FN}}\rangle\, \| \leq \epsilon_{FN}$, and ϵ^2_{FN} is the (small) *probability of false negative*; and in the negative (non shifted) case, it maps:

$$|\psi^t_{f,g}\rangle\,|b\rangle \xmapsto{\mathsf{DAHS}} |\psi^t_{f,g}\rangle\,|b\rangle + |\delta^{f,g}_{\mathrm{FP}}\rangle\,|b \oplus 1\rangle$$

where $\| \,|\delta^{f,g}_{\mathrm{FP}}\rangle\, \| \leq \epsilon_{FP}$ and ϵ^2_{FP} is the (small) *probability of false positive*. We now bound both ϵ_{FN} and ϵ_{FP} separately, and independently of f and g.

False Negatives. False negatives come from all the sequences of labels $Y = y_1, \ldots, y_t$ on which the combination cannot produce enough labels 2^{n-1}. As unitary operators preserve the euclidean norm, we can bound $\| \, |\delta_{\mathrm{FN}}\rangle \, \|$ before the combination step. Let us consider the state:

$$|\phi_{f,g}\rangle^{\otimes t} = \sum_X \sum_Y \chi_M(X \cdot Y) \left(\bigotimes_{1 \leq i \leq t} (|0\rangle + \chi_M(-y_i s) \, |1\rangle) \, |y_i\rangle \right) |f(X)\rangle \quad , \quad (1)$$

where $X = x_1, \ldots, x_t$, $\chi_M(X \cdot Y) = \prod \chi_M(x_i y_i)$ and $f(X) = f(x_1), \ldots, f(x_t)$. All the "bad" sequences Y contribute to the probability of false negatives, so we bound:

$$\left\| \sum_X \sum_{Y \text{ bad}} \chi_M(X \cdot Y) \Big(\bigotimes_i (|0\rangle + \chi_M(-y_i s) \, |1\rangle) \, |y_i\rangle \Big) \, |f(X)\rangle \right\| \leq \sqrt{\frac{|\text{bad } Ys|}{|\text{all } Ys|}} \quad .$$

Thus, *if both f and g are injective*, we have $\epsilon_{FN} = \sqrt{\frac{|\text{bad } Ys|}{|\text{all } Ys|}}$ (and this does not depend from f and g). The classical analysis of Kuperberg's algorithm gives a constant probability of finding a good label if we start from $\mathcal{O}\left(2^{\sqrt{2 \log_2 3n}}\right)$ of them. Thus, if we take copies of the combination circuit, we can obtain r good labels with probability $1 - \epsilon$ if we start from $\mathcal{O}\left((-\log_2 \epsilon) r 2^{\sqrt{2 \log_2 3n}}\right)$ labels.

False Positives. We have seen above that when the functions are not shifted, we have still some probability to fall on $|0^r\rangle$ or $|1^r\rangle$ after the final Hadamard transform. This probability is also independent of f and g. The amplitude on the state $|0^r\rangle$ or $|1^r\rangle$ is exactly $\frac{1}{\sqrt{2^r}}$, and for all f: $\| \, |\delta_{FP}\rangle \, \|^2 = \frac{2}{2^r} \implies \epsilon_{FP} = 2^{-(r-1)/2}$.

5 The Offline-DAHS Algorithm

In this section, we describe an offline-DAHS algorithm that will help us solve the SLS with classical queries, based on the reversible circuit DAHS.

The algorithm looks for a pair of shifted functions $g(\cdot) = f(\cdot + s)$ over an abelian group, when g is fixed and f goes through a family $(f_i)_{i \in I}$. We will consider a simple version of this problem, in which the group is $\mathbb{Z}_{2^n}$, all functions are injective and admit distinct image sets.

Problem 5 (Finding a shifted pair, injective case). Let $g : \mathbb{Z}_{2^n} \to X$ be a function, and $f_i : \mathbb{Z}_{2^n} \to X$ be a family of functions indexed by I, such that: • g and all f_i are injective; • there exists a single $i_0 \in I$ and a shift s such that: $\forall x \in \mathbb{Z}_{2^n}, g(x) = f_{i_0}(x + s)$; • $\bigcup_{i \neq i_0} Im(f_i)$ and $Im(g)$ are disjoint. Then find i_0.

5.1 High-level Description

First of all, we describe offline-DAHS in a generic way, following the layout of the offline-Simon algorithm by Bonnetain et al. [8]. Note that [8] already proposed to combine their algorithm with Kuperberg's, but did not analyze the resulting algorithm nor its time complexity. Originally, the offline-Simon algorithm, which itself follows Grover-meet-Simon [30], combines a quantum search and a quantum circuit for Simon's algorithm. Roughly speaking, offline-DAHS is obtained by replacing Simon's algorithm by Kuperberg's (which is why we needed to define a quantum circuit for it).

Description. We use a quantum search for the right index $i_0 \in I$. Testing a given i means finding whether f_i is a shift of g. For this, we use the circuit DAHS. Recall that DAHS takes in input the sample database of (f_i, g): $|\psi_{f_i,g}\rangle^{\otimes t}$ and writes a single output bit. Thus, the naive Grover search would reconstruct the database at each iteration, then compute DAHS, then return the database to $|0\rangle$.

However, due to the asymmetric nature of the problem, the function g in the database remains the same from one iteration to the next. Let us introduce the sample database of $(0, g)$:

$$|\psi_{0,g}\rangle^{\otimes t} = \left(\sum_x |0\rangle\,|x\rangle\,|0\rangle + |1\rangle\,|x\rangle\,|g(x)\rangle \right)^{\otimes t}.$$

It contains all the data on g that we need for the successive iterations of quantum search. The algorithm has then two steps:

- Precomputation step: Construct $|\psi_{0,g}\rangle^{\otimes t}$.
- Search step: Run the quantum search for i_0. At each search iterate, compute f_i inside the database to obtain the state $|\psi_{f_i,g}\rangle^{\otimes t}$, run the circuit DAHS, then compute f_i again to return to the state $|\psi_{0,g}\rangle^{\otimes t}$. Due to the *approximate* nature of DAHS, this is a quantum search with an approximate test (Theorem 1).

We have bounded the error of DAHS and given its complexity in Theorem 2. As long as the initial state $|\psi_{0,g}\rangle^{\otimes t}$ can be constructed exactly, this becomes an easy instance of Theorem 1, and we deduce:

Proposition 1. *Let $\alpha = 2\log_2 3$. Problem 5 can be solved within a time $\widetilde{\mathcal{O}}(2^{\sqrt{\alpha n}}\sqrt{I})$ using $\widetilde{\mathcal{O}}(2^{\sqrt{\alpha n}})$ qubits, with constant probability. There are $\widetilde{\mathcal{O}}(2^{\sqrt{\alpha n}})$ queries to g and $\widetilde{\mathcal{O}}(2^{\sqrt{\alpha n}}\sqrt{I})$ queries to f (in superposition for both).*

The fact that the queries to O_g are now performed only in the precomputation step is the reason for the *offline* denomination. Note that the polynomial factors in the $\widetilde{\mathcal{O}}$ in Proposition 1 are not negligible, and depend on $\log_2 I$ and n together.

5.2 Approximate Promise

We now go into technical details specific to offline-DAHS and not discussed in the previous literature.

So far, our analysis has assumed that we could start from an *exact* sample database $|\psi_{0,g}\rangle^{\otimes t}$ as defined above. But it is not the case in the SLS problem. Here, g will be the secretly shifted Legendre symbol $\left(\frac{s+\cdot}{P}\right)$. This function is defined on $\mathbb{Z}_P$, but it is queried on an interval of length M.

In particular, in order to run the algorithm as expected from Proposition 1, we would need sample states of the form:

$$|\psi_{0,g}^{\text{exact}}\rangle = \sum_{0\leq x\leq M-1} |0\rangle\,|x\rangle\,|0\rangle + \sum_{-s\leq x\leq M-s-1} |1\rangle\,|x\rangle\,|g(x)\rangle \quad ,$$

which would allow for a total interference between the matching values of $f(x)$ and $g(x)$, when querying the good f.

However, since we do not know the value of s, such states cannot be created. Instead, we will rely on *approximate* sample states:

$$|\psi_{0,g}^{\text{approx}}\rangle = \sum_{0\leq x\leq M-1} |0\rangle\,|x\rangle\,|0\rangle + \sum_{0\leq x\leq M-1} |1\rangle\,|x\rangle\,|g(x)\rangle \quad .$$

There is an error vector $|\psi_{0,g}^{\text{err}}\rangle$ such that $|\psi_{0,g}^{\text{approx}}\rangle = |\psi_{0,g}^{\text{exact}}\rangle + |\psi_{0,g}^{\text{err}}\rangle$:

$$\| \, |\psi_{0,g}^{\text{err}}\rangle \, \| = \left\| \sum_{-s\leq x\leq -1} |1\rangle\,|x\rangle\,|g(x)\rangle - \sum_{M-s\leq x\leq M-1} |1\rangle\,|x\rangle\,|g(x)\rangle \right\| \leq \sqrt{\frac{2s}{4M}} \quad ,$$

and we can bound the distance between the "exact" database of $(0, g)$ and the "approximate" one:

$$\left\| \left(|\psi_{0,g}^{\text{approx}}\rangle\right)^{\otimes t} - \left(|\psi_{0,g}^{\text{exact}}\rangle\right)^{\otimes t} \right\|^2 \leq t \| \, |\psi_{0,g}^{\text{err}}\rangle \, \|^2 = \frac{ts}{2M} \quad .$$

This gives a total "starting error" $\nu = \sqrt{\frac{ts}{2M}}$. As we have seen in Theorem 1, we need this error to be constant, thus s needs to be subexponentially smaller than M for the algorithm to work.

5.3 Application to the Legendre Symbol

Given a sequence of $M = 2^n$ successive outputs $\left(\frac{s+x}{P}\right)$ of the Legendre PRF, we define a function $g(x) = L_{s+x}$ on $\mathbb{Z}_M$. Next, we choose an integer n' such that $2^{n'} < M$ and we write: $s = s_1 + 2^{n'} s_2$, where $s_1 < 2^{n'}$. For a given s_2, we define $f(x) = L_{x+2^{n'} s_2}$. Then there exists a single s_2 such that $g(x) = f(x + s_1)$.

Note that we need a subexponential gap between $2^{n'}$, where n' is the number of bits of the secret handled by the DAHS subroutine, and $M = 2^n$, the amount of data given. It is due to the approximation discussed in Section 5.2. The DAHS subroutine still runs with labels of $n = \log_2 M$ bits.

By taking L-sequences of length $\geq 3\log_2 P$, our Heuristic 1 ensures that the functions are injective, and that they have distinct image sets: two L-sequences cannot collide randomly. Thus, we have an instance of Problem 5 with an approximate promise, and we can apply Theorem 1.

We use the M classical queries to build the approximate sample states $|\psi_{0,g}^{\text{approx}}\rangle$, and then, we run a Grover search over the remaining secret s_2. Building the sample database from the classical queries is costly ($\mathcal{O}(Mm)$ quantum gates for each sample), but done only once in the precomputation step.

Let $\alpha = 2\log_2 3$. In order to make the starting error ν smaller than $\frac{1}{4}$, we must take t labels where $\sqrt{t2^{n'}/(2M)} \leq \frac{1}{4} \implies t \leq M/2^{n'+3}$. But since $t = \widetilde{\mathcal{O}}(2^{\sqrt{\alpha n}})$, this means the circuit DAHS can only recover n' bits of s, where $n' + \sqrt{\alpha n} = \log_2 M = n$. Thus $n' = n - \sqrt{\alpha n}$. The remaining $(\log_2 P - n')$ bits must be searched with Grover's algorithm.

Theorem 3. *Given a sequence of M outputs of the Legendre PRF, we can solve the SLS problem using $\widetilde{\mathcal{O}}\left(2^{\sqrt{\alpha\log_2 M}}\right)$ qubits, in quantum time:*

$$\widetilde{\mathcal{O}}\left(M2^{\sqrt{\alpha\log_2 M}}\right) + \widetilde{\mathcal{O}}\left(2^{\frac{3}{2}\sqrt{\alpha\log_2 M}}\sqrt{\frac{P}{M}}\right) \quad .$$

Proof. We use Corollary 1 and Theorem 2. We run a quantum search with an approximate test (the circuit DAHS) and an approximate starting state (the approximate sample states).

The analysis of DAHS assumes an exact ancillary state $|\psi_{0,g}\rangle^{\otimes t}$. Thus, Theorem 1 is crucial here to ensure that the approximate starting state does not disrupt the algorithm. □

The minimum occurs when the two terms are equal, which constrains: $\sqrt{\alpha\log_2 M} + \log_2 M = \frac{3}{2}\sqrt{\alpha\log_2 M} + \frac{1}{2}\log_2 P - \frac{1}{2}\log_2 M \implies \log_2 M = \frac{1}{3}\sqrt{\alpha\log_2 M} + \frac{1}{3}\log_2 P$, up to the polynomial factors. We get $\sqrt{\log_2 M} = \sqrt{\frac{1}{3}\log_2 P + \frac{\alpha}{36} + \frac{\sqrt{\alpha}}{6}} = \sqrt{\frac{1}{3}\log_2 P} + \mathcal{O}(1)$ and $\log_2 M = \frac{1}{3}\log_2 P + \frac{\sqrt{\alpha}}{3\sqrt{3}}\sqrt{\log_2 P} + \mathcal{O}(1)$. This gives a time complexity of order: $P^{1/3}2^{\frac{4}{3}\sqrt{\alpha}\sqrt{\log_2 P^{1/3}}}$.

6 Conclusion

In this paper, we presented two quantum algorithms for solving the Legendre hidden shift problem (SLS) when classical queries are given, and without quantum RAM. The first one (distinguished table-based collisions) allows to reach an advantage against Grover's algorithm when more data is given. The second one is the *offline Kuperberg's algorithm*. It is an interesting method, notably the only one reaching a time-memory product below $\mathcal{O}\left(\sqrt{P}\right)$. However, some numerical estimations (see Appendix 11) suggest that its subexponential factor renders it impractical so far.

While all algorithms studied in this paper can be applied to the Ethereum challenges, in which a sequence of successive Legendre symbols is given, they do not concern all applications of the Legendre PRF. In the signature scheme LegRoast [6], the adversary learns only the evaluation of the PRF on a few random inputs. In that case, the best classical and quantum attacks are table-based collision searches with small tables, in time $\widetilde{\mathcal{O}}(P)$ and $\widetilde{\mathcal{O}}\left(\sqrt{P}\right)$ respectively.

Although a very efficient algorithm exists when superposition queries are allowed [18], it does not seem amenable to an *offline* version, which requires to define reduced instances of the problem (e.g., guessing part of the secret and finding the remaining bits by searching for a shift). Nevertheless, the algebraic properties of the Legendre symbol might still find a use in this context, and we leave this as an open question.

Acknowledgments. A.S. would like to thank Xavier Bonnetain for discussions on Kuperberg's algorithm. Both authors would like to thank Jeeun Lee, Changmin Lee and the anonymous reviewers of Mathcrypt for helpful comments. This work has been supported by the European Union's H2020 project No. 714294 (QUASYModo) and by ERC-ADG-ALSTRONGCRYPTO (project 740972).

References

1. Ambainis, A.: Quantum walk algorithm for element distinctness. SIAM J. Comput. 37(1), 210–239 (2007)
2. Batcher, K.E.: Sorting networks and their applications. In: AFIPS Spring Joint Computing Conference. AFIPS Conference Proceedings, vol. 32, pp. 307–314. Thomson Book Company, Washington D.C. (1968)
3. Beals, R., Brierley, S., Gray, O., Harrow, A.W., Kutin, S., Linden, N., Shepherd, D., Stather, M.: Efficient distributed quantum computing. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 469(2153), 20120686 (2013)
4. Bennett, C.H., Bernstein, E., Brassard, G., Vazirani, U.V.: Strengths and weaknesses of quantum computing. SIAM J. Comput. 26(5), 1510–1523 (1997)
5. Beullens, W., Beyne, T., Udovenko, A., Vitto, G.: Cryptanalysis of the legendre PRF and generalizations. IACR Trans. Symmetric Cryptol. 2020(1), 313–330 (2020)
6. Beullens, W., de Saint Guilhem, C.D.: Legroast: Efficient post-quantum signatures from the legendre PRF. In: PQCrypto. Lecture Notes in Computer Science, vol. 12100, pp. 130–150. Springer (2020)
7. Bonnetain, X.: Improved low-qubit hidden shift algorithms. CoRR abs/1901.11428 (2019), http://arxiv.org/abs/1901.11428
8. Bonnetain, X., Hosoyamada, A., Naya-Plasencia, M., Sasaki, Y., Schrottenloher, A.: Quantum attacks without superposition queries: The offline simon's algorithm. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 11921, pp. 552–583. Springer (2019)
9. Bonnetain, X., Naya-Plasencia, M.: Hidden shift quantum cryptanalysis and implications. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 11272, pp. 560–592. Springer (2018)

10. Bonnetain, X., Schrottenloher, A.: Quantum security analysis of CSIDH. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 12106, pp. 493–522. Springer (2020)
11. Brassard, G., Hoyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. Contemporary Mathematics 305, 53–74 (2002)
12. Brassard, G., Høyer, P., Tapp, A.: Quantum cryptanalysis of hash and claw-free functions. In: LATIN. Lecture Notes in Computer Science, vol. 1380, pp. 163–169. Springer (1998)
13. Brent, R.P., Zimmermann, P.: An $O(M(n)\ \log n)$ algorithm for the jacobi symbol. In: ANTS. Lecture Notes in Computer Science, vol. 6197, pp. 83–95. Springer (2010)
14. Chailloux, A., Naya-Plasencia, M., Schrottenloher, A.: An efficient quantum collision search algorithm and implications on symmetric cryptography. In: ASIACRYPT (2). Lecture Notes in Computer Science, vol. 10625, pp. 211–240. Springer (2017)
15. Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Zaverucha, G.: Post-quantum zero-knowledge and signatures from symmetric-key primitives. In: CCS. pp. 1825–1842. ACM (2017)
16. Childs, A.M., Jao, D., Soukharev, V.: Constructing elliptic curve isogenies in quantum subexponential time. J. Math. Cryptol. 8(1), 1–29 (2014)
17. Cuccaro, S.A., Draper, T.G., Kutin, S.A., Moulton, D.P.: A new quantum ripple-carry addition circuit. arXiv preprint quant-ph/0410184 (2004)
18. van Dam, W., Hallgren, S., Ip, L.: Quantum algorithms for some hidden shift problems. SIAM J. Comput. 36(3), 763–778 (2006)
19. Damgård, I.: On the randomness of legendre and jacobi sequences. In: CRYPTO. Lecture Notes in Computer Science, vol. 403, pp. 163–172. Springer (1988)
20. Feist, D.: Legendre pseudo-random function (2019), https://legendreprf.org, accessed: 2021-01-19
21. Grassi, L., Rechberger, C., Rotaru, D., Scholl, P., Smart, N.P.: Mpc-friendly symmetric key primitives. In: CCS. pp. 430–443. ACM (2016)
22. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: STOC. pp. 212–219. ACM (1996)
23. Helm, A., May, A.: The power of few qubits and collisions - subset sum below grover's bound. In: PQCrypto. Lecture Notes in Computer Science, vol. 12100, pp. 445–460. Springer (2020)
24. Jaques, S., Schanck, J.M.: Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE. In: CRYPTO (1). Lecture Notes in Computer Science, vol. 11692, pp. 32–61. Springer (2019)
25. Kaluderovic, N., Kleinjung, T., Kostic, D.: Improved key recovery on the legendre PRF. IACR Cryptol. ePrint Arch. 2020, 98 (2020)
26. Kaluđerović, N., Kleinjung, T., Kostić, D.: Cryptanalysis of the generalised legendre pseudorandom function. In: ANTS. Open Book Series, vol. 4, pp. 267–282. Mathematical Sciences Publishers (2020)
27. Khovratovich, D.: Key recovery attacks on the legendre PRFs within the birthday bound. IACR Cryptol. ePrint Arch. 2019, 862 (2019)
28. Kuperberg, G.: A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. SIAM J. Comput. 35(1), 170–188 (2005)
29. Kuperberg, G.: Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem. In: TQC. LIPIcs, vol. 22, pp. 20–34. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2013)
30. Leander, G., May, A.: Grover meets simon - quantumly attacking the fx-construction. In: ASIACRYPT (2). Lecture Notes in Computer Science, vol. 10625, pp. 161–178. Springer (2017)

31. Nielsen, M.A., Chuang, I.: Quantum computation and quantum information (2002)
32. Peikert, C.: He gives c-sieves on the CSIDH. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 12106, pp. 463–492. Springer (2020)
33. Regev, O.: A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space. arXiv preprint quant-ph/0406151 (2004)
34. Roetteler, M., Naehrig, M., Svore, K.M., Lauter, K.E.: Quantum resource estimates for computing elliptic curve discrete logarithms. In: ASIACRYPT (2). Lecture Notes in Computer Science, vol. 10625, pp. 241–270. Springer (2017)
35. Seres, I.A., Horváth, M., Burcsi, P.: The legendre pseudorandom function as a multivariate quadratic cryptosystem: Security and applications. IACR Cryptol. ePrint Arch. 2021, 182 (2021), https://eprint.iacr.org/2021/182
36. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: FOCS. pp. 124–134. IEEE Computer Society (1994)

APPENDIX

7 Detailed Complexities

We give in Table 2 a detailed version of Table 1.

Table 2. Comparison of classical and quantum algorithms to solve the SLS problem, including previous works and our new results. We note $m = 3\lceil \log_2 P \rceil$, $\alpha = 2\log_2 3$, $L \leq (\log_2 P)^2$ the time to compute a Legendre symbol, and we omit constant factors.

Method	Queries	Time	Memory	Source
		Classical algorithms		
Pollard's rho	$\sqrt{P}m$	$L\sqrt{P}m$	m	[27]
Table	M	$M^2 + Pm^2/M^2$	M^2/m	[5]
Table	M	$M^2 + Pm\log_2 m/M^2$	M^2	[25]
		Quantum algorithms		
Sup. queries	2	$\mathsf{poly}(m)$	$\mathsf{poly}(m)$	[18]
Table (QRACM)	M	$M^2 + m^2\sqrt{P/M^2}L$	M^2 QRACM	[25]
Grover search	m	$\log_2 m\sqrt{P}L$	m qubits	Sec. 3.2
Distinguished points	M	$M^2 + \frac{\sqrt{P}}{M^{1/3}}m^{\frac{11}{6}}\log_2\log_2 M$	M classical + m qubits	Sec. 3.3
Offline-DAHS	M	$\widetilde{O}\left(\begin{array}{c} M2^{\sqrt{\alpha\log_2 M}} + \\ 2^{\frac{3}{2}\sqrt{\alpha\log_2 M}}\sqrt{\frac{P}{M}} \end{array}\right)$	$\widetilde{O}(2^{\sqrt{\alpha\log_2 M}})$ qubits	Sec. 5.3

8 Proof of Theorem 2

In this section, we prove Theorem 1.

Let $\mathcal{A}, a, t = \left\lfloor \frac{\pi}{4\theta_a} \right\rfloor$ be defined as in Theorem **??**. Let f be a boolean function that tests if an output of $\mathcal{A}$ is "good", and let O'_f be an approximate oracle for f with error ϵ, using an ancillary state $|\psi\rangle$, as per Definition 1. Let $|\psi'\rangle$ be a quantum state such that $\| \, |\psi\rangle - |\psi'\rangle \, \| \leq \nu$. We run the following algorithm:

$$\left((\mathsf{I} \otimes \mathcal{A})(\mathsf{I} \otimes O_0)(\mathsf{I} \otimes \mathcal{A}^\dagger)O'_f\right)^t |\psi'\rangle \quad ,$$

that is, t iterations of "approximate" quantum search using O'_f, where I is the identity operator applied to the ancillary state $|\psi\rangle$. Then measuring the output yields a good result with probability greater than $(1 - t\epsilon - \nu)^2 \max(1 - a, a)$.

Proof. The proof uses a "hybrid argument" as in [4] or [1, Lemma 5]. We will consider the "perfect" run of the algorithm, that starts with the initial $|\psi\rangle$ and applies the perfect test O_f, and compare it with the "imperfect" one, that starts with $|\psi'\rangle$ and applies the imperfect test O'_f.

Let $|\psi'_k\rangle$ be the state after k iterations of the imperfect search, and $|\psi_k\rangle$ after the perfect search. Our goal is to bound $\| \, |\psi'_k\rangle - |\psi_k\rangle \, \|$. Let $U = (\mathsf{I} \otimes \mathcal{A})O_0(\mathsf{I} \otimes \mathcal{A}^\dagger)$, then the quantum search iterates are respectively UO'_f and UO_f. Before the first iteration, we have:

$$\| \, |\psi'_0\rangle - |\psi_0\rangle \, \| = \nu \quad ,$$

by definition of $|\psi'\rangle$. Next, for each $k \geq 0$:

$$\begin{aligned} \left\| |\psi'_{k+1}\rangle - |\psi_{k+1}\rangle \right\| &= \left\| UO'_f \, |\psi'_k\rangle - UO_f \, |\psi_k\rangle \right\| \\ &= \left\| O'_f \, |\psi'_k\rangle - O_f \, |\psi_k\rangle \right\| \\ &= \left\| O'_f(|\psi'_k\rangle - |\psi_k\rangle) + O'_f \, |\psi_k\rangle - O_f \, |\psi_k\rangle \right\| \quad , \end{aligned}$$

and using the triangle inequality:

$$\begin{aligned} \left\| |\psi'_{k+1}\rangle - |\psi_{k+1}\rangle \right\| &\leq \left\| O'_f(|\psi'_k\rangle - |\psi_k\rangle) \right\| + \left\| O'_f \, |\psi_k\rangle - O_f \, |\psi_k\rangle \right\| \\ &\leq \| |\psi'_k\rangle - |\psi_k\rangle \| + \left\| O'_f \, |\psi_k\rangle - O_f \, |\psi_k\rangle \right\| \quad . \end{aligned}$$

In order to bound the second term, we use the fact that O'_f induces a *uniform* error ϵ. More specifically, if $|\psi_k\rangle = \sum_x \alpha_x \, |x\rangle$, we have: $O'_f \, |\psi_k\rangle - O_f \, |\psi_k\rangle = \sum_x \alpha_x \, |\delta_x\rangle$ where $|\delta_x\rangle$ is the error induced by O'_f on the input state x. Then we have

$$\left\| O'_f \, |\psi_k\rangle - O_f \, |\psi_k\rangle \right\|^2 = \sum_x \alpha_x^2 \, \| |\delta_x\rangle \|^2 \leq \epsilon^2 \implies \left\| O'_f \, |\psi_k\rangle - O_f \, |\psi_k\rangle \right\| \leq \epsilon \quad ,$$

by definition of ϵ. Thus, each iteration adds an error ϵ.

After t iterations, we have $|\psi'_t\rangle = |\psi_t\rangle + |\psi_{\text{err}}\rangle$ where $\| |\psi_{\text{err}}\rangle \| \leq \nu + t\epsilon$. By the Cauchy-Schwarz inequality, we have:

$$| \, \langle \psi_t | \psi_{\text{err}} \rangle \, | \leq \| |\psi_t\rangle \| \, \| |\psi_{\text{err}}\rangle \| \leq \nu + t\epsilon \quad .$$

Measuring $|\psi'_t\rangle$, we project on $|\psi_t\rangle$ with a probability greater than:

$$(1 - |\langle\psi_t|\psi_{\text{err}}\rangle|)^2 \geq (1 - \nu - t\epsilon)^2$$

and then, by Theorem **??**, we have a probability greater than $\max(1-a, a)$ to measure a "good" element. □

It can seem surprising to require a *uniform* error ϵ. Indeed, in a classical exhaustive search with a single solution, we can afford a constant probability of false negative (not recognizing the solution), and still obtain a constant probability of success, as we expect to look at the solution only once.

Our classical intuition then dictates that the same should be true of a quantum search: we could afford a much higher probability of false negatives than of false positives. We found that this was not the case, due to the stateful nature of the search. Indeed, taking different ϵ_{FP} and ϵ_{FN} changes the error term ϵ added at iteration k to a term:

$$\sqrt{\sin^2((2k+1)\theta_a)\epsilon_{FN}^2 + \cos^2((2k+1)\theta_a)\epsilon_{FP}^2} \ .$$

In order to have a constant probability of success in the end, we must measure a state in which a constant proportion of the amplitude is on the solution. That is, $(2k+1)\theta_a$ must be sufficiently close to $\frac{\pi}{2}$ to have $\sin^2((2k+1)\theta_a)$ constant. This means that in the last iterations, the error term is close to ϵ_{FN}. (Although in the first iterations, it was close to ϵ_{FP}). Over all the iterations, the solution and the bad elements both capture roughly (up to a constant) the same amount of amplitude, and so both terms ϵ_{FN} and ϵ_{FP} will have roughly the same effect.

9 Quantum Circuit for the Legendre Symbol

The prime P is fixed. We detail a quantum circuit that given x, computes $\left(\frac{x}{P}\right)$. We will actually adopt the more general view of computing Jacobi symbols. We recall Algorithm 1 that uses the multiplicativity, the law of quadratic reciprocity and its supplement:

$$\forall p, q, a, b, \left(\frac{q}{p}\right)\left(\frac{p}{q}\right) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}, \quad \left(\frac{q}{p}\right) = \left(\frac{q \bmod p}{p}\right),$$
$$\left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}}, \quad \left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right) \ .$$

The situation is similar to the extended GCD algorithm studied in [34]: in the classical algorithm, the number of steps depends on the input. The corresponding quantum circuit must run for a fixed amount of iterates, thus we take the greatest possible number of iterates and control them depending on whether the algorithm has finished or not. Note that there exists asymptotically more efficient classical algorithms, such as [13], but it is not clear whether they can have an impact on the design of quantum circuits (especially for rather small prime numbers).

Algorithm 1 Computation of the Legendre symbol, through the Jacobi symbol.

```
1: function LEGENDRE(q,p)
   Computes: (q/p) where p is prime and 1 ≤ q < p
2:     t = 1
3:     while q ≠ 0 do
4:         if q mod 2 = 0 then
5:             q = q/2
6:             if p mod 8 = 3 or p mod 8 = 5 then
7:                 t = −t
8:             end if
9:         else if q < p then
10:            q, p = p, q
11:            if p mod 4 = 3 and q mod 4 = 3 then
12:                t = −t
13:            end if
14:            q = q − p          ▷ actually q = q mod p in the standard algorithm
15:        else
16:            q = q − p
17:        end if
18:    end while
19:    return t
20: end function
```

Let us analyze briefly Algorithm 1. We can easily prove that if q does not start even, then in the next loop it is. Thus, the current pair (p, q) is reduced by one bit at each two loops, and after *at most* $2(\lceil \log_2 p \rceil + \lceil \log_2 q \rceil) \leq 4 \lceil \log_2 p \rceil$ loop iterations, the computation of $\left(\frac{q}{p}\right)$ terminates.

In order to compute the Legendre symbol modulo P, we thus use a circuit of $4 \lceil \log_2 P \rceil$ iterations. We keep a "flag" qubit indicating whether the computation has finished, and a "result" qubit containing the value t of Algorithm 1. At each iteration, we look at the "flag", the values of p and q and find which case applies:

- Case 1: we must divide q by 2 ($q > 0$ and $q \mod 2 = 0$)
- Case 2: we must swap q and p ($q > 0, q \leq p$ and $q \mod 2 = 1$)
- Case 3: we must subtract p to q ($q > 0$, $q > p$ and $q \mod 2 = 1$)
- Case 4: we must do nothing ($q = 0$)

Two new qubits indicate the case; their computation requires a comparator ($\mathcal{O}(\log_2 P)$ gates). Then we apply all the operations controlled on these qubits. The division by 2 is implemented as a rotation of the register (we know that the least significant bit is 0, so we do not need to write a carry). The swap is an easy operation. The subtraction does not need any carry either. Then t is flipped depending on the case (this is also reversible, and costs a constant number of computations).

Since each iteration requires $\mathcal{O}(\log_2 P)$ gates and writes $\mathcal{O}(1)$ new qubits, the complete circuit requires $\mathcal{O}\left((\log_2 P)^2\right)$ gates and $\mathcal{O}(\log_2 P)$ ancilla qubits.

10 Combination Circuit

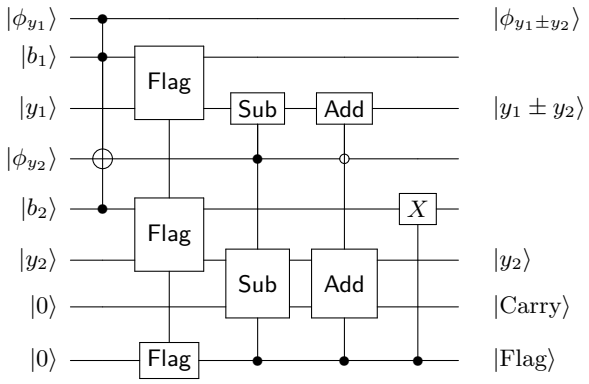

Fig. 1. Combination circuit $\mathsf{Combine}_v$ from Section 4.4. Note the "control on 0" which applies the addition only if the qubit $|\phi_{y_2}\rangle$ is 0.

11 Simulations and Cost Estimates

In this section, we give more precise, non-asymptotic (but heuristic) complexity estimates for offline-DAHS applied to the SLS problem. Our goal is to determine for which values of P the algorithm becomes more efficient than the table-based collision search. We expect that for small P, its advantage in the exponential term will be defeated by the subexponential factor.

Note that for exhaustive search and table-based collisions, the asymptotic formulas of Table 2 are very close to the non-asymptotic costs (only by a small constant factor).

11.1 Number of Labels in DAHS

As we have seen, the number of labels dictates the probability of false positives and false negatives in the DAHS circuit. This is a property of the combination step, independent from the functions f, g that we are actually studying.

We will first fix $r = 200$, meaning that we want the combination step to produce 200 labels equal to 2^{n-1}. This reduces the probability of false positives to $2^{-(r-1)} = 2^{-199}$.

Next, we will estimate how many initial label states we need to reduce the probability of false negatives below 2^{-199} as well. We upper bound this probability by the probability that a given sequence of random labels $y_1, \ldots, y_t$, drawn at random, is "bad", i.e., does not allow to produce 200 labels equal to

2^{n-1}. Concretely, we choose a value of t and we sample sequences of t initial labels uniformly at random. We then perform the combination step as in the DAHS circuit, i.e., with layers of combination circuits and of sorting.

During this step, we actually simulate the output of DAHS when starting from the sequence of labels $y_1, \ldots, y_t$. This is doable because the entire circuit (except the initial QFTs, which produce the labels in the first place) contains only classical reversible operations such as CNOTs and controlled additions. We can then check how many labels equal to 2^{n-1} are produced.

Our goal is still to estimate the probability:

$$\Pr_{y_1,\ldots,y_t \xleftarrow{\$} \mathbb{Z}_{2^n}} \left(y_1, \ldots, y_t \text{ produce } r \text{ labels equal to } 2^{n-1} \right) .$$

However, we cannot estimate it experimentally, since we need it to be extremely small.

Heuristic. The heuristic assumption that we make is that the number of labels equal to 2^{n-1}, that the sieve produces, follows a normal distribution. Our experiments allow then to estimate the mean and standard deviation. We estimate the probability to obtain less than 200 labels by integrating the density function on $]-\infty; 200]$.

Since the circuit is better layout with $t = 2^\ell$, we increase the number of labels in powers of 2, and there is always a clear threshold. At some point, the average number of good labels exceeds 700 and the probability of failure becomes negligibly smaller than 2^{-200} (by our estimate).

We display some simulation results in Table 3 for different values of n and ℓ. In general, if there are enough labels for a given n, then the same circuit should work for $n' < n$. This is not always what we observe (see the column $(60, 19)$), because having a smaller n' can *increase* the number of zero-labels produced, to the detriment of good labels. In that case we must take a smaller number of labels, but we can layout the circuit as if there were 2^ℓ of them exactly (with dummy labels).

Table 3. Estimates of the number of labels required (in $\log_2$). The numbers are rounded to the nearest integer. We highlight the values of n, ℓ for which we estimate a probability of failure (having less than 200 labels) smaller than 2^{-200}.

n	36	**36**	40	**40**	**44**	**48**	**52**	**56**	60	**64**	**68**	**72**	**76**
ℓ	16	**17**	17	**18**	**18**	**18**	**18**	**19**	19	**19**	**20**	**20**	**21**
Mean	361	1517	408	1259	728	1374	933	1289	565	811	1601	1199	2307
Std.	17	36	19	33	28	37	26	38	21	26	38	29	45

11.2 Comparison

Let us take an example of SLS parameters: $\log_2 P = 240$ and $\log_2 M = 80$. The best classical table-based collision reaches a complexity roughly 2^{120}, as

does quantum exhaustive search. The quantum table-based collision will strictly improve on that, but we can leave it aside for now.

We will run the DAHS circuit with labels in $\mathbb{Z}_M$, thus 80 bits. By our estimates in Appendix 11.1, in order to have negligible errors in the DAHS circuit, we can use $t = 2^{21}$ labels for $\log_2 M = 80$. Thus we need 2^{21} sample states.

The DAHS circuit will only recover n' bits of the secret, where $n' < 80$ depends on the bound on the error ν, that we need to make smaller than $\frac{1}{4}$. This constrains:

$$\sqrt{\frac{t2^{n'}}{2M}} \leq \frac{1}{4} \implies n' \leq \log_2 M - \log_2 t - 3 = 56 \ .$$

Building a sample state requires roughly M register-wise operations; this makes 2^{101} in total for the first step of offline-DAHS. Next, since there remains $\log_2 P - n' = 184$ bits of secret, the quantum search requires $\frac{\pi}{4}2^{92}$ iterations. But each of them costs at least $80 \times 21^2 \times 2^{21} = 2^{36}$ n-qubit operations, due to the 80 layers of sorting in DAHS. This brings the total complexity above exhaustive search.

Thus, the subexponential factor of Kuperberg's sieve, but also the polynomial factors induced by our reversible circuit, defeat the time speedup for a prime P of 300 bits. With the circuit that we presented, the improvements for larger values of P (400 or 500 bits) will likely remain outperformed by the table-based collision search.

In order to overturn this situation, the circuit DAHS must be made more efficient. There are different possible ways:

- Increasing the resistance of the search to the errors in DAHS. Having to produce a large number of labels is problematic for small parameters.
- Using a more involved variant of Kuperberg's algorithm, for example Kuperberg's collimation sieve [29]. It is currently unclear to us if its asymptotic improvement could also translate into a non-asymptotic improvement of the circuit layout.
- Reducing the overall cost of the sorting and combination layers. It is clear that most of the work in the final layers is useless, since most of the labels have already been consumed. It might be possible to reduce this cost using a refined estimation on the number of remaining labels at each step.