Quantum Security of the Legendre PRF

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

[1] Orange Labs, Caen, France
[2] Inria, Paris, France

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 exhaustive search of the key. The first one is a quantum variant of the table-based collision algorithm. The second one uses Kuperberg's abelian hidden shift algorithm in an *offline* manner. We show that the latter, although asymptotically promising, is not currently the most efficient against practical parameters.

Keywords: Legendre PRF, quantum cryptanalysis, quantum algorithms, 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 [15], who conjectured the hardness of the following problem:

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

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

That is, the Legendre symbol produces 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 [14]:

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

or as in [17], by remapping $\{-1, 1\}$ on $\{0, 1\}$. For a given 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 recovery of the secret s. At the moment, no separation between the SLS and the DSLS is known, which is why we (as all previous works) focus on the SLS.

Problem 2 (Shifted Legendre symbol). Let P be a prime number. Given query access to the function $F_{\mathrm{Leg},s} : x \mapsto \left(\frac{x+s}{P}\right)$ 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.* [17] to use it in a multi-party computation scenario. Indeed, the malleability of the Legendre symbol allows to evaluate the PRF very efficiently in this setting. 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 [16].

Previous Results. The SLS problem can be seen as a particular case of the *shifted character problem*, which was studied by van Dam *et al.* [14] in the quantum setting. When *superposition query access* to the shifted character is given (in our case, the Legendre PRF $F_{\mathrm{Leg},s}$), they showed that the problem could be solved in polynomial time and with only a single query to $F_{\mathrm{Leg},s}$.

However, when only *classical* queries are given, the SLS problem was conjectured by van Dam *et al.* and by Grassi *et al.* to remain intractable for a quantum attacker. In the classical setting, several authors have studied and improved the Legendre PRF key-recovery attacks [20, 19, 4]. We give a summary in Table 1. The most advanced results were obtained by Beullens *et al.* [4] and concurrently by Kaluderović *et al.* [19]. 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.

Contributions. We perform the first analysis of quantum algorithms to solve the SLS without quantum random-access memories (we note that an algorithm with quantum RAM was proposed in [19], and that some remarks on quantum security were made in [4]). We give two techniques and we discuss their applicability: first, a *quantum table-based collision search* that builds over the multi-target preimage search of [11]. Second, an *offline abelian hidden shift algorithm*, which is an offline adaptation of Kuperberg's algorithm [21], similar to the *offline Simon's algorithm* of Bonnetain *et al.* [6]. It was already mentioned in [6] that Kuperberg's algorithm could undergo the same "offline" treatment as Simon's, but this had never been studied before in full detail. We show that the Legendre PRF provides a nice application for this. However, although it is asymptotically more efficient, we conclude that the offline Kuperberg's algorithm would require significant improvements to be competitive with the quantum table-based collision search. Our results are summarized in Table 1.

Organization of the Paper. We give some technical preliminaries in Section 2. In Section 3, we recall the classical table-based attacks and introduce our first

Method	Queries	Time	Memory	Source
		Classical algorithms		
Pollard's rho	$\sqrt{P}m$	$L\sqrt{P}m$	m	[20]
Table	M	$M^2 + Pm^2/M^2$	M^2/m	[4]
Table	M	$M^2 + Pm\log_2 m/M^2$	M^2	[19]
		Quantum algorithms		
Sup. queries	2	$\mathsf{poly}(m)$	$\mathsf{poly}(m)$	[14]
Table (qRAM)	M	$M^2 + m^2\sqrt{P/M^2}L$	M^2 qRAM	[19]
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 Kuperberg	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

Table 1. Comparison of classical and quantum algorithms to solve the SLS problem. 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.

algorithm. In Section 4, we recall Kuperberg's first algorithm and introduce its reversible version (most technical details are deferred to Appendix D). In Section 5, we introduce the *offline Kuperberg's algorithm* and apply it to the SLS problem.

2 Preliminaries

In this section, we introduce some preliminaries of quantum computing.

2.1 Quantum Circuits

We refer to [23] for an introduction to the quantum circuit model. Our quantum algorithms are described as quantum circuits: the time complexity is counted as the number of basic quantum gates used (*e.g.* Clifford + T gates) and the space complexity is the number of qubits. Sometimes, also, we count the *classical memory* used. As our considerations are mainly asymptotic, most implementation details remain out of scope of our work. For example, we consider that a modular addition of two n-bit values costs $\mathcal{O}(n)$ basic gates [13]. Swapping two n-qubit registers, comparing two n-qubit registers, or comparing a given n-qubit register with a classical value all cost $\mathcal{O}(n)$ gates. We show in Appendix C that the Legendre symbol modulo P can be computed with $\mathcal{O}\left((\log_2 P)^2\right)$ basic gates.

We use the term *ancilla* qubits for qubits that are initialized to the state $|0\rangle$, used by the circuit and returned to their initial state. We say that a computation

is performed *out of place* if its output is written on an additional register and *in place* otherwise. We routinely use the standard "ket" notation of quantum states $|\psi\rangle$, the notation O_f for a quantum oracle for f and standard operators such as the M-dimensional Quantum Fourier Transform:

$$|x\rangle \xmapsto{\mathrm{QFT}_M} \frac{1}{\sqrt{M}} \sum_{y \in \mathbb{Z}_M} \exp(2i\pi/M)^{xy} |y\rangle \quad .$$

2.2 Quantum Search

In this paper, we make use of Amplitude Amplification [9], a powerful generalization of Grover's algorithm [18]. It allows to speed up the search for a "good" output of *any* probabilistic algorithm, including another quantum algorithm.

Theorem 1 ([9], Theorem 2). *Let $\mathcal{A}$ be a quantum algorithm that uses no measurements, let $f : X \to \{0,1\}$ be a boolean function that tests if an output of $\mathcal{A}$ is "good". Let a be the success probability of $\mathcal{A}$. Let O_0 be the "inversion around zero" operator that does: $O_0 |x\rangle = (-1)^{x \neq 0} |x\rangle$ and O_f a quantum oracle for f: $O_f |x\rangle = (-1)^{f(x)} |x\rangle$. Let $\theta_a = \arcsin \sqrt{a}$ and $0 < \theta_a \leq \frac{\pi}{2}$. Let $t = \left\lfloor \frac{\pi}{4\theta_a} \right\rfloor$. Then by measuring $(\mathcal{A}O_0\mathcal{A}^\dagger O_f)^t \mathcal{A} |0\rangle$, we obtain a good result with success probability greater than $\max(1-a, a)$.*

Next, we use another result on Grover search using an *approximate test oracle*. This notion was introduced in [6]. In this context, the test oracle O_f admits an *ancillary state* $|\psi\rangle$ which must be preserved from one iteration to the next: $|x\rangle |\psi\rangle \xmapsto{O_f} (-1)^{f(x)} |x\rangle |\psi\rangle$. However, we have only an *imperfect* oracle O'_f, that induces a uniform error of amplitude ϵ: $\forall x, |x\rangle |\psi\rangle \xmapsto{O'_f} (-1)^{f(x)} |x\rangle |\psi\rangle + |\delta_x\rangle$ where $\max_x \| |\delta_x\rangle \| \leq \epsilon$. In addition to [6], we will also start from an *approximate initial state*.

Theorem 2 (Grover search with approximate test (see [6])). *Consider the setting of Theorem 1 with an approximate test O'_f of error ϵ. On an input $|\psi'\rangle \otimes (\mathcal{A} |0\rangle)$, where $\| |\psi\rangle - |\psi'\rangle \| \leq \nu$, we run $t = \left\lfloor \frac{\pi}{4\theta_a} \right\rfloor$ iterations of Grover search with O'_f. Then measuring the output yields a good result with probability greater than $(1 - t\epsilon - \nu)^2 \max(1-a, a)$.*

We give a proof in Appendix B. The idea of the proof is a "hybrid argument" as in [3] or [1, Lemma 5]. Each iteration with the approximate test adds a global error of amplitude less than ϵ, and we start with an error ν. The state after t iterations differs from the "ideal" one by an error of amplitude less than $t\epsilon + \nu$. Upon measurement, we project on the "ideal" state with probability greater than $(1 - t\epsilon - \nu)^2$ and we then measure a good element with probability $\max(1-a, a)$. Note that the fact that ν can be constant, as it is "paid" only once, is particularly important for us.

Corollary 1. *If $\nu \leq \frac{1}{4}$ and $\epsilon \leq \frac{1}{4t}$, then the procedure of Theorem 2 succeeds with probability $\geq \frac{1}{8}$.*

3 Table-based Attacks

In this section, we recall the classical *table-based collision search* of [4, 19], which is the basis of the *quantum table-based collision search* presented after. Note that we follow the presentation in [4].

3.1 Classical Algorithm

Let $m = 2\lceil \log_2 P \rceil$. The attack uses queries to the function $\left(\frac{x+s}{P}\right)$ to create *L-sequences*: words of m bits that allow to discriminate a guess of s. The definition of L-sequences is from [4] 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", a collision of L-sequences implies the equality of their parameters. With sequences of size $m = 3\lceil \log_2 P \rceil$, 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, which 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 s, one then looks for a collision between an L-sequence of unknown parameter (depending on the secret s) and an L-sequence of known parameter. With M data, we can obtain $M - m$ L-sequences, but the attack of [4] uses the multiplicativity of the Legendre symbol to increase significantly this number. Assume that $M < \sqrt{P}$ consecutive values of $\left(\frac{x+s}{P}\right)$ are known, then the attack does:

1. Extract $\frac{M^2}{m}$ L-sequences of the form:

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

   Naively, we would have extracted only M sequences from the data, but the multiplicativity of the Legendre symbol allows to extract $\frac{M^2}{m}$ sequences, using $b \leq \left\lfloor \frac{M}{m} \right\rfloor$ and $a < M - bm + 1$.
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 is 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.

This classical attack requires M^2 memory of 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 qRAM. This procedure yields a quantum attack as proposed in [19]. The precomputation stage (Step 1) is unchanged, but now Step 3 is a Grover 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 contains $\mathcal{O}\left(m\right)$ Legendre symbol computations and a memory access. However, the memory requires quantum random-access (qRAM), a powerful model that we do not consider in this paper.

3.2 Grover Search

With $M < \sqrt{P}$ queries available, and without quantum RAM, the best quantum attack available is a direct Grover search of the secret s. We query the L-sequence L_s (thus using only m data) and search for the single $x \in \mathbb{Z}_p^*$ (by Heuristic 1) such that $L_x = L_s$. This first version requires $\mathcal{O}(m\sqrt{P})$ Legendre symbol computations, thus $\mathcal{O}(m^3\sqrt{P})$ 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. This folklore idea allows to save *almost* a factor m on Grover search: instead of searching for x such that $L_x = L_s$, we define subsets of $\mathbb{Z}_P^*$: $X_i = \{x \in \mathbb{Z}_P^*, L_x \text{ and } L_s \text{ match on the first } i \text{ bits }\}$. Our goal is to find an element of X_m, but as we have an inclusion $X_1 \subseteq X_2 \subseteq \ldots \subseteq X_m$, we will search for X_m *only in* X_i for some $i \leq m$. Assuming that $|X_i| \simeq \frac{P}{2^i}$, this gives a complexity: $\frac{\pi}{4}\sqrt{\frac{P}{2^i}}\left(2\frac{\pi}{4}\sqrt{2^i}(iL) + (m-i)L\right)$, where we use a Grover search to sample from X_i in time $\frac{\pi}{4}\sqrt{2^i}(iL)$, inside an Amplitude Amplification. We can see that taking $i = \log_2 m$ reduces the total complexity to $\mathcal{O}(\log_2 m\sqrt{P}L)$.

3.3 Distinguished Collisions

Since we do not assume qRAM, 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 [11] 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 c such that L_c is distinguished, until it matches one of the stored sequences.

We use Amplitude Amplification. Sampling a distinguished L_c is done in time $\mathcal{O}(\log_2 t\sqrt{2^t}\ell)$ with an early-aborted Grover search. Testing whether L_c matches our table is done with a sequential circuit: since all the values are known classically, $\mathcal{O}(m)$ controlled quantum gates allow to compare a single L_a to the current L_c. Assuming that we use all the data, this yields an algorithm of complexity:

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

Note that the number of iterations has been reduced, because the probability of two distinguished sequences to collide is higher than two random sequences. By taking $t = \frac{4}{3}\log_2(M/m)$ we get a complexity:

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

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. Contrary to the classical table-based collision, which can use up to $M = P^{/4}$ queries, this one will stop improving at $M = P^{3/14}$ queries, where it reaches $\tilde{\mathcal{O}}\left(P^{3/7}\right)$.

4 Kuperberg's Algorithm and Its Reversible Version

In this section, we recall the *abelian hidden shift problem* and Kuperberg's first algorithm [21]. Its standard depiction highly depends on intermediate measurements. Our new contribution is to describe a *reversible* variant of the algorithm, study its time complexity and error probability, and give some simulation results.

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 [27] 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 the *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 [14], the SLS problem is an instance of Problem 3, where g will be the PRF $F_{\mathrm{Leg},s}$ and f will be the Legendre symbol modulo P. However, if quantum oracle access to g is allowed, the dedicated algorithm of [14] is much more efficient. 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

In [21], 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 [25, 22, 12, 5, 24, 8]. 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* [22]. In this paper, we focus on the earliest algorithm, not only because it is easier to present, but also because the other versions are *hybrid* algorithms, which tend to replace quantum costs by classical costs. For example, the variant of Regev [25] requires to solve classical subset-sum instances. While this is very useful if Kuperberg's algorithm is used as a standalone procedure, it helps less if we make it a *fully reversible quantum circuit*, which is our goal here.

In this paper, we assume that the underlying abelian group is $G = \mathbb{Z}_{2^n}$ for some n. We let $M = 2^n$ denote the cardinality of the group. Note that the generalization to an arbitrary abelian group is technical, but without a significant incidence on the time complexity [8]. The assumption that the functions are injective is also helpful and sufficient for our study, but not strictly necessary.

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

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

where the common amplitude factor is omitted. It is easy to produce such a state with a single oracle query to quantum oracles for f and 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).

Label States. Each of these states is then transformed into a *label state* by measuring a value a in the second register and applying an M-dimensional QFT on the third register. 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:

$$|\phi_{f,g}\rangle = \mathrm{QFT}_M(|0\rangle\,|x_0\rangle + |1\rangle\,|x_0 - s\rangle)$$

$$= \sum_{0 \leq y \leq M-1} \left(\chi_M(x_0 y)\,|0\rangle + \chi_M((x_0 - s)y)\,|1\rangle\right)|y\rangle$$
$$= \sum_{0 \leq y \leq M-1} \chi_M(x_0 y)(|0\rangle + \chi_M(-sy)\,|1\rangle)\,|y\rangle$$

where we define $\chi_M(z) = \exp\left(\frac{2\pi z}{M}\right)$. The value of a is discarded. Next, we can measure the register $|y\rangle$ to obtain a random value y and a *label qubit*: $|\phi_y\rangle = |0\rangle + \chi_M(-sy)\,|1\rangle$ (the global phase factor does not matter).

Such a 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 $|\phi_{2^{n-1}}\rangle$. The bulk of the algorithm is then the *combination step*, that allows to create the wanted labels from the random initial ones.

Combining two Labels. Starting from two label qubits $|\phi_{y_1}\rangle$ and $|\phi_{y_2}\rangle$, we can obtain a label qubit for $y_1 \pm y_2$, that is $|\phi_{y_1+y_2}\rangle$ with probability $\frac{1}{2}$, and $|\phi_{y_1-y_2}\rangle$ otherwise. Starting from the joint state:

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

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

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

and we then measure the second qubit. Up to a global phase factor (which does not matter), we obtain either $|\phi_{y_1+y_2}\rangle$ or $|\phi_{y_1-y_2}\rangle$. This operation destroys the qubits $|\phi_{y_1}\rangle$ and $|\phi_{y_2}\rangle$.

The Classical Procedure. The procedure consists in combining pairs of labels (y_1, y_2) having the same valuation modulo 2, that maximize the expected valuation of $y_1 \pm y_2$. Interestingly, it can be described and analyzed classically. In particular, it is very easy to simulate its behavior by sampling the labels at random, as done in [7].

Complexity. The complexity analysis in [21] gives that $\widetilde{\mathcal{O}}\left(2^{\sqrt{(2\log_2 3)n}}\right)$ initial label qubits are enough. Simulations in [7] showed that $2^{\sqrt{(2\log_2 3)n}}$ were essentially enough to solve Problem 3 with constant probability.

4.3 Reversible Variant

The principle of *deferred measurements* states that measurements are not necessary in a quantum algorithm, and they can simply be not performed. Thus, there exists a quantum circuit DAHS that solves Problem 4. However, the standard procedure contains non-trivial memory operations which, without quantum RAM, may increase significantly the time complexity in a fully reversible variant. We show in Appendix D that reversibility costs only a polynomial factor in time. An exponentially small failure probability can be obtained with another polynomial factor.

Theorem 3. *There exists a quantum circuit* DAHS *that maps:* $|b\rangle |\psi_{f,g}\rangle^{\otimes t}$ *to* $|b \oplus 1\rangle |\psi_{f,g}\rangle^{\otimes t}$ *or* $|b\rangle |\psi_{f,g}\rangle^{\otimes t}$*, where the bit* b *is flipped if and only* f *is a shift of* g*. With* $t = \widetilde{\mathcal{O}}\left(2^{\sqrt{2\log_2 3n}}\right)$*, it contains* $\widetilde{\mathcal{O}}\left(2^{\sqrt{2\log_2 3n}}\right)$ *quantum gates and ancilla qubits. It has a constant probability of error, that can be reduced to* 2^{-m} *with a factor* m *in time and memory complexity.*

5 The Offline Kuperberg's Algorithm

In this section, we describe the *offline Kuperberg's algorithm* based on the reversible circuit DAHS. Its goal is to find 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, similarly to Problem 4, 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

We follow the layout of the *offline Simon's algorithm* by Bonnetain *et al.* [6]. We use Grover's algorithm to 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, 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. Thus, we introduce the sample database of $(0, g)$: $|\psi_{0,g}\rangle^{\otimes t} = \left(\sum_x |0\rangle |x\rangle |0\rangle + \sum_x |1\rangle |x\rangle |g(x)\rangle\right)^{\otimes t}$.

The algorithm then has two steps:

- Precomputation step: Construct the sample database of $(0, g)$: $|\psi_{0,g}\rangle^{\otimes t}$.

- Search step: Run the Grover 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 Grover search with an approximate test (Theorem 2).

The queries to g are now made only in the precomputation step, in an *offline* manner. They require no additional storage, since they can be simply consumed on the fly while constructing $|\psi_{0,g}\rangle^{\otimes t}$.

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).*

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. We provide more details in Appendix D.

5.2 Approximate Promise

In order to use this algorithm to solve the SLS problem, we need to adapt the promise. The input group of the functions will now be $\mathbb{Z}_P$ (since these are Legendre symbols modulo P). However, they are still queried on an interval of length M. In order to run the algorithm as before, we would need sample states of the form:

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

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 rely on *approximate* sample states:

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

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

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

Since there are t such sample states in the database, the total "starting" error is: $\nu := \sqrt{\frac{ts}{2M}}$. Thus we will need s to be subexponentially smaller than M for the algorithm to work.

5.3 Relation to the Legendre Symbol

We will first show how to solve the SLS problem with the offline Kuperberg's algorithm. Given a sequence of M 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 here, there will be a subexponential gap between 2^n (n being the number of bits of the secret handled by the DAHS subroutine) and M (the amount of data given). It is due to the approximation discussed in Section 5.2 above.

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

Theorem 4. *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:*

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

The minimum occurs for $M \simeq P^{1/3}$ and reaches $\widetilde{\mathcal{O}}\left(2^{\frac{4}{3}\sqrt{\alpha \log_2 P}} P^{1/3}\right)$.

In Practice. Let us take an example: $\log_2 P = 240$ and $\log_2 M = 80$. The best classical table-based collision reaches a complexity roughly 2^{120}. In the DAHS circuit, we will use 2^ℓ labels and recover n bits, where $(\ell + n - \log_2 M - 1) \leq -4$. By our estimates in Appendix D.4, in order to have negligible errors in the DAHS circuit, we can fit $n = 58$ and use 2^{19} labels. Thus we need 2^{19} sample states, each of which requires roughly M register-wise operations, thus 2^{99} in total. Since $\log_2 P - n = 182$, only $\frac{\pi}{4} 2^{91}$ iterations of Grover search are required. But each of them costs at least $58 \times 19^2 \times 2^{19} \simeq 2^{32}$ n-qubit operations, due to the 58 layers of sorting in DAHS, thus bringing the total complexity above the classical one.

Due to these constraints, the algorithm does not seem fit to achieve quantum time speedups for a prime P of 300 bits. The improvements for higher values

(400 or 500 bits) will likely remain outperformed by the distinguished collision attack of Section 3.3.

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*. Our analysis showed that although it is asymptotically promising, its subexponential factor renders it impractical so far. But with a refined circuit layout, optimizations and more precise estimates, some improvements might be possible.

Although a very efficient algorithm exists when superposition queries are allowed [14], 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. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 714294 - acronym QUASYModo).

References

[1] Ambainis, A.: Quantum walk algorithm for element distinctness. SIAM J. Comput. 37(1), 210–239 (2007), https://doi.org/10.1137/S0097539705447311

[2] Batcher, K.E.: Sorting networks and their applications. In: American Federation of Information Processing Societies: AFIPS Conference Proceedings: 1968 Spring Joint Computer Conference, Atlantic City, NJ, USA, 30 April - 2 May 1968. AFIPS Conference Proceedings, vol. 32, pp. 307–314. Thomson Book Company, Washington D.C. (1968), https://doi.org/10.1145/1468075.1468121

[3] Bennett, C.H., Bernstein, E., Brassard, G., Vazirani, U.V.: Strengths and weaknesses of quantum computing. SIAM J. Comput. 26(5), 1510–1523 (1997)

[4] Beullens, W., Beyne, T., Udovenko, A., Vitto, G.: Cryptanalysis of the legendre PRF and generalizations. IACR Trans. Symmetric Cryptol. 2020(1), 313–330 (2020), https://doi.org/10.13154/tosc.v2020.i1.313-330

[5] Bonnetain, X.: Improved low-qubit hidden shift algorithms. CoRR abs/1901.11428 (2019), http://arxiv.org/abs/1901.11428

[6] Bonnetain, X., Hosoyamada, A., Naya-Plasencia, M., Sasaki, Y., Schrottenloher, A.: Quantum attacks without superposition queries: The offline simon's algorithm. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019

- 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11921, pp. 552–583. Springer (2019), https://doi.org/10.1007/978-3-030-34578-5_20

[7] Bonnetain, X., Naya-Plasencia, M.: Hidden shift quantum cryptanalysis and implications. In: Peyrin, T., Galbraith, S.D. (eds.) Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11272, pp. 560–592. Springer (2018), https://doi.org/10.1007/978-3-030-03326-2_19

[8] Bonnetain, X., Schrottenloher, A.: Quantum security analysis of CSIDH. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12106, pp. 493–522. Springer (2020), https://doi.org/10.1007/978-3-030-45724-2_17

[9] Brassard, G., Hoyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. Contemporary Mathematics 305, 53–74 (2002)

[10] Brent, R.P., Zimmermann, P.: An $O(M(n)\ \log n)$ algorithm for the jacobi symbol. In: Hanrot, G., Morain, F., Thomé, E. (eds.) Algorithmic Number Theory, 9th International Symposium, ANTS-IX, Nancy, France, July 19-23, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6197, pp. 83–95. Springer (2010), https://doi.org/10.1007/978-3-642-14518-6_10

[11] Chailloux, A., Naya-Plasencia, M., Schrottenloher, A.: An efficient quantum collision search algorithm and implications on symmetric cryptography. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10625, pp. 211–240. Springer (2017), https://doi.org/10.1007/978-3-319-70697-9_8

[12] Childs, A.M., Jao, D., Soukharev, V.: Constructing elliptic curve isogenies in quantum subexponential time. J. Math. Cryptol. 8(1), 1–29 (2014), https://doi.org/10.1515/jmc-2012-0016

[13] 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)

[14] van Dam, W., Hallgren, S., Ip, L.: Quantum algorithms for some hidden shift problems. SIAM J. Comput. 36(3), 763–778 (2006), https://doi.org/10.1137/S009753970343141X

[15] Damgård, I.: On the randomness of legendre and jacobi sequences. In: Goldwasser, S. (ed.) Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings. Lecture Notes in Computer Science, vol. 403, pp. 163–172. Springer (1988), https://doi.org/10.1007/0-387-34799-2_13

[16] Feist, D.: Legendre pseudo-random function (2019), https://legendreprf.org, accessed: 2021-01-19

[17] Grassi, L., Rechberger, C., Rotaru, D., Scholl, P., Smart, N.P.: Mpc-friendly symmetric key primitives. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016. pp. 430–443. ACM (2016), https://doi.org/10.1145/2976749.2978332

[18] Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Miller, G.L. (ed.) Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996. pp. 212–219. ACM (1996), https://doi.org/10.1145/237814.237866
[19] Kaluderovic, N., Kleinjung, T., Kostic, D.: Improved key recovery on the legendre PRF. IACR Cryptol. ePrint Arch. 2020, 98 (2020), https://eprint.iacr.org/2020/098
[20] Khovratovich, D.: Key recovery attacks on the legendre PRFs within the birthday bound. IACR Cryptol. ePrint Arch. 2019, 862 (2019), https://eprint.iacr.org/2019/862
[21] Kuperberg, G.: A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. SIAM J. Comput. 35(1), 170–188 (2005), https://doi.org/10.1137/S0097539703436345
[22] Kuperberg, G.: Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem. In: Severini, S., Brandão, F.G.S.L. (eds.) 8th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2013, May 21-23, 2013, Guelph, Canada. LIPIcs, vol. 22, pp. 20–34. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2013), https://doi.org/10.4230/LIPIcs.TQC.2013.20
[23] Nielsen, M.A., Chuang, I.: Quantum computation and quantum information (2002)
[24] Peikert, C.: He gives c-sieves on the CSIDH. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12106, pp. 463–492. Springer (2020), https://doi.org/10.1007/978-3-030-45724-2_16
[25] Regev, O.: A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space. arXiv preprint quant-ph/0406151 (2004)
[26] Roetteler, M., Naehrig, M., Svore, K.M., Lauter, K.E.: Quantum resource estimates for computing elliptic curve discrete logarithms. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10625, pp. 241–270. Springer (2017), https://doi.org/10.1007/978-3-319-70697-9_9
[27] Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994. pp. 124–134. IEEE Computer Society (1994), https://doi.org/10.1109/SFCS.1994.365700
[28] Simon, D.R.: On the power of quantum computation. SIAM J. Comput. 26(5), 1474–1483 (1997), https://doi.org/10.1137/S0097539796298637

Appendix

A The Shifted Character Algorithm

In this section, we review the algorithm of [14] for the shifted character problem, that uses superposition queries. This is a polynomial-time algorithm, thus much more efficient than Kuperberg's. Similarly to Simon's algorithm [28], it relies on Fourier sampling. But the algebraic properties of characters also play a major role.

We present it in the case of finite fields $\mathbb{Z}_P$ with P prime, although it works in any finite field.

Multiplicative and Additive Characters. Multiplicative characters are functions: $\chi : \mathbb{Z}_P^* \to \mathbb{C}^*$ such that $\chi(xy) = \chi(x)\chi(y)$ for all x and y. They are extended to $\mathbb{Z}_p$ by setting $\chi(0) = 0$. Furthermore $\chi(x)$ is a complex number of norm 1. For any such character χ, there exists a generator g of $\mathbb{Z}_P$ and some k such that $\chi(g^\ell) = \omega_{P-1}^{k\ell}$, where $\omega_P = \exp(2i\pi/P)$. The Legendre symbol is an example of such a character.

Problem 6 (Shifted character). Let χ be a multiplicative character of $\mathbb{Z}_P$. Given access to a function f such that $f(x) = \chi(x + s)$ for some s, find s.

Additive characters of $\mathbb{Z}_P$ are functions: $\psi : \mathbb{Z}_P \to \mathbb{C}^*$ such that $\psi(x + y) = \psi(x)\psi(y)$.

Computing in Amplitudes. The characters that we consider are complex-valued functions which return either 0, or a complex number of modulus 1. Thus, it is possible to compute them *in the amplitude*: $\sum_x f(x)\,|x\rangle$ instead of $\sum_x |x\rangle\,|f(x)\rangle$. This is done in [14] as follows: starting with $\sum_x |x\rangle$, we find the superposition such that $f(x) \neq 0$ (assuming that $f(x) = 0$ almost never happens, which will be the case). Then we compute the phase of $f(x)$ into the phase.

Algorithm 1 Shifted multiplicative character algorithm from [14].

Input: query access to $x \mapsto \chi(x + s)$
Output: s
1: Query: $\sum_{x\in\mathbb{Z}_P} \chi(x + s)\,|x\rangle$
2: Compute the Fourier transform over $\mathbb{Z}_P$
3: Compute $\chi(y)$ into the phase
4: Compute the inverse Fourier transform over $\mathbb{Z}_P$
5: Measure $-s$

Algorithm 1 is the algorithm of [14]. We detail its steps to prove its correctness. At Step 2 the state becomes:

$$\sum_x \left(\sum_y \chi(x + s)\omega_P^{xy}\,|y\rangle \right) = \sum_y \left(\sum_x \chi(x + s)\omega_P^{xy} \right) |y\rangle \quad .$$

In the following, we actually reprove a well-known property of Gauss sums. We do a change of variables $z = (x+s)y$:

$$\sum_x \chi(x+s)\omega_P^{xy} = \sum_z \chi(z/y)\omega_P^{z-sy} = \chi(y)^{-1}\omega_P^{-sy}\sum_z \chi(z)\omega_P^z \ .$$

Hence, we can discard the sum $\sum_z \chi(z)\omega_P^z$ as a common amplitude factor and rewrite the state after Step 2 as:

$$\sum_y \chi(y)^{-1}\omega_P^{-sy}\,|y\rangle \ .$$

At Step 3, we compute $\chi(y)$ into the phase, canceling the factor $\chi(y)^{-1}$:

$$\sum_y \omega_P^{-sy}\,|y\rangle \ .$$

And finally, we compute an inverse Fourier transform, mapping this to $|-s\rangle$.

The Limits of an Offline Variant. The offline Simon's algorithm of [6] and the offline Kuperberg's algorithm that we studied in this paper share the same structure: the problem of recovering the shift is reduced to a smaller one, with an exhaustive search on the rest of the secret.

In Simon's and Kuperberg's algorithm, only an additive group structure is used, and it enables to embed such a reduced problem. However, the efficiency of Algorithm 1 comes from using both a multiplicative and an additive structure. This is why it does not seem to admit an offline variant. This does not mean, however, that the multiplicative properties of the Legendre symbol cannot be used to improve over the offline Kuperberg's algorithm. But this remains an open question.

B Proof of Theorem 2

In this section, we prove Theorem 2.

> Consider the setting of Theorem 1 with an approximate test O'_f of error ϵ. On an input $|\psi'\rangle\otimes(\mathcal{A}\,|0\rangle)$, where $\|\,|\psi\rangle-|\psi'\rangle\,\| \leq \nu$, we run $t = \left\lfloor \frac{\pi}{4\theta} \right\rfloor$ iterations of Grover search with O'_f. 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 [3] 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, together 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:

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

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

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

and using the triangle inequality:

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

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

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

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 1, 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. Indeed, 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 each iteration to a term:

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

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.

C 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 2 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 in [26]: 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 [10], but it is not clear whether they can have an impact on the design of quantum circuits (especially for rather small prime numbers).

Let us analyze briefly Algorithm 2. 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 2. 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.

Algorithm 2 Computation of the Legendre symbol, through the Jacobi symbol.

function LEGENDRE(q,p)
  Computes: $\left(\frac{q}{p}\right)$ where p is prime and $1 \leq q < p$
  $t = 1$
  while $q \neq 0$ **do**
    if $q \bmod 2 = 0$ **then**
      $q = q/2$
      if $p \bmod 8 = 3$ or $p \bmod 8 = 5$ **then**
        $t = -t$
      end if
    else if $q < p$ **then**
      $q, p = p, q$
      if $p \bmod 4 = 3$ and $q \bmod 4 = 3$ **then**
        $t = -t$
      end if
      $q = q - p$ ▷ actually $q = q \bmod p$ in the standard algorithm
    else
      $q = q - p$
    end if
  end while
  return t
end function

D A Quantum Circuit for Kuperberg's Algorithm

In this section, we give the details omitted from Section 4 and describe our quantum circuit DAHS in detail. We also give detailed simulation results.

D.1 Combining two Labels Reversibly

As we have seen in Section 4, the bulk of the algorithm is the combination step. We start from a pool of label qubits and we combine them pairwise.

Combination Subroutine. The registers that contain the labels remain unmeasured. Thus, when combining two labels, we start from the joint state:

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

and afterwards, 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 get to the state $(|\phi_{y_1+y_2}\rangle|0\rangle + \chi_M(-(y_1-y_2)s)|\phi_{y_1-y_2}\rangle|1\rangle)|y_2\rangle$, we add or subtract y_2 in place, on the register for y_1, controlled on the qubit that we previously discarded.

In our DAHS circuit, we will define a "combination circuit" $\mathsf{Combine}_v$. We flag all the label qubits with additional qubits that inform us whether the label can be used for combination or not. The circuit $\mathsf{Combine}_v$ then combines the two labels if and only if both have valuation v and can be combined. It performs a controlled addition or subtraction in place, modifies the flag qubits (since one of the labels cannot be used for combination anymore), and writes a carry qubit.

Choosing which Labels to Combine. Among the pairs of labels y_1, y_2 having the same valuation v modulo 2 ($v = \mathrm{val}_2(y_1)$), 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[3]:

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

where val_2 is the valuation and $\mathrm{rev}(y, n)$ 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.

D.2 Combining All Labels

We start from a list of $t = 2^\ell$ labels 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 $\mathsf{Combine}_v$ 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 combination 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 as to ensure a maximal number of zeroes after combination, since we take them 2 by 2 on arbitrary positions

[3] Although we did not find its explicit definition in previous works, it appears in the simulation code of [7].

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 "junk" 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, to know if the algorithm succeeded, it suffices to test the m first registers for equality to 2^{n-1}.

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 $\mathsf{CombineAll}_{2^\ell,m}$ 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 $\mathcal{O}\left(n^2 S(2^\ell)\right) = \mathcal{O}\left(n^2\ell^2 2^\ell\right)$ quantum gates used, mainly for comparators and SWAPs. The circuit writes $nS(2^\ell) + (n-1)2^\ell$ ancillas. They are uncomputed afterwards.

If the sequence of initial labels can be combined into m copies of 2^{n-1}, then after the layer $n-2$, the m first label qubits in the circuit contain copies of $|0\rangle \pm |1\rangle$. We perform a Hadamard transform on them and test if the result is all-zero or all-one. We write a pair of qubits $|\mathrm{Bad}(Y), \mathrm{Result}(Y)\rangle$ that indicate if the combination succeeded and if so, what is the result of the test. Then, we uncompute the combination layers.

If $|Y\rangle$ is the initial label state, then the circuit $\mathsf{CombineAll}_{2^\ell,m}$ maps:

$$|Y\rangle\,|0,0\rangle \xmapsto{\mathsf{CombineAll}} |Y\rangle\,|\mathrm{Bad}(Y), \mathrm{Result}(Y)\rangle \quad .$$

If the combination did not succeed, then the result cannot be trusted, and this will be a source of errors in the circuit DAHS.

D.3 Errors in the Full Circuit

Recall that we input a state $|\psi_{f,g}\rangle^{\otimes t}\,|b\rangle$ to the circuit DAHS, where $|\psi_{f,g}\rangle^{\otimes t}$ is the *sample database* that must be kept unchanged, and $|b\rangle$ is a qubit to write the result $\mathsf{DAHS}(f,g)$:

$$\mathsf{DAHS}(f,g) = \begin{cases} 1 \text{ if } \exists s, f(x+s) = g(x) \text{ (positive case)} \\ 0 \text{ if } f \text{ and } g \text{ have no image in common (negative case)} \end{cases} .$$

Layout. We set $t = \widetilde{\mathcal{O}}\left(2\sqrt{2\log_2 3n}\right)$. We start by applying a QFT on all sample states, obtaining the label states:

$$|\phi_{f,g}\rangle^{\otimes t} = \sum_{x_1,\ldots,x_t} \sum_{y_1,\ldots,y_t} \chi_M\left(\sum_i x_i y_i\right) \Big(\left(|0\rangle + \chi_M(-y_1 s)\,|1\rangle\right)|y_1\rangle \ldots$$

$$\dots\left(|0\rangle+\chi_M(-y_2 s)\,|1\rangle\right)\dots\left(|0\rangle+\chi_M(-y_t s)\,|1\rangle\right)|y_t\rangle\Bigg)\,|f(x_1)\dots f(x_t)\rangle \quad . \quad (4)$$

Next, we apply the circuit $\mathsf{CombineAll}_{t,m}$. If $|\text{Bad}\rangle$ is set, write 0 in the output. Otherwise we write $|\text{Result}\rangle$ in the output. Next, we uncompute $\mathsf{CombineAll}_{t,m}$ and we uncompute the QFT.

Errors. The circuit DAHS is not exact. In the positive case, it maps:

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

where $\|\,|\delta_{\text{FN}}\rangle\,\| \leq \epsilon_{FN}$, and ϵ_{FN} is the (small) *probability of false negative*; and in the negative case, it maps:

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

where $|\delta^{f,g}_{\text{FP}}\rangle$ may depend on f and g, and $\|\,|\delta_{\text{FP}}\rangle\,\| \leq \epsilon_{FP}$ is the (small) *probability of false positive*. We now bound both ϵ_{FN} and ϵ_{FP}.

False Negatives. False negatives comes from all the sequences of labels Y on which the combination fails. As unitary operators preserve the euclidean norm, we can bound $\|\,|\delta_{\text{FN}}\rangle\,\|$ before the QFTs and the combination step. In the state:

$$|\phi_{f,g}\rangle^{\otimes t} = \sum_X \sum_Y \chi_M(X\cdot Y)\left(\bigotimes_{1\leq i\leq t}\left(|0\rangle+\chi_M(-y_i s)\,|1\rangle\right)|y_i\rangle\right)|f(X)\rangle\,, \quad (5)$$

all the "bad" sequences Y contribute to the probability of false negatives, and we bound:

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

Thus, *if both f and g are injective*, we have $\epsilon_{FN} = \sqrt{\frac{|\text{bad } Ys|}{|\text{all } Ys|}}$. 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 m good labels with probability $1-\epsilon_{FN}$ if we start from $\mathcal{O}\left(-\log_2\epsilon_{FN} m 2^{\sqrt{2\log_2 3n}}\right)$ labels. We give more precise estimates in Section D.4.

False Positives. When the functions are not shifted, the output $|\text{Result}\rangle$ of the combination step will still be 1 in some cases. Indeed, while the combination produces "random" qubits, we are merely going to test if m of them are 1 or 0 altogether. This still happens with some probability.

We assume that the images of f and g are distinct subsets of the codomain X. Thus, after the QFT, the current state of the circuit is a superposition of vectors of the form $|Y\rangle|i\rangle|\phi_i\rangle$ where Y is the sequence $y_1, \ldots, y_t$ of labels, $|i\rangle$ is the aggregation of the qubits that did not undergo the QFTs, and $|\phi_i\rangle$ is an injective function of i and $y_1, \ldots, y_t$, that corresponds to a sequence of images by f or g. But since these images are independent, the basis states $|Y\rangle|i\rangle$ cannot interfere with each other: the combination layer only swaps these states without changing their amplitude, which remain uniform. Thus, regardless of the combinations of labels performed, the amplitude on the state $|0^m\rangle$ or $|1^m\rangle$ is exactly $\frac{1}{\sqrt{2^n}}$, and for all f: $\| |\delta_{FP}\rangle \|^2 = \frac{2}{2^m} \implies \epsilon_{FP} = 2^{-(m-1)/2}$.

D.4 Estimates of the Number of Labels

As we have seen, in the DAHS circuit, we need to choose the right number of labels if we are to ensure a small probability of false negatives. We will now make precise estimates for typical values of n, picturing "practical" applications of the circuit[4]. We fix $m = 200$ in what follows.

n	**20**	32	**32**	36	**36**	40	**40**	**44**	**48**	**52**	**56**	60	**64**	**68**	**72**	**76**
ℓ	**15**	15	**16**	16	**17**	17	**18**	**18**	**18**	**18**	**19**	19	**19**	**20**	**20**	**21**
Samples	10^3	10^3	10^3	10^3	10^3	10^3	50	50	50	50	50	50	50	50	50	50
Mean	810	399	1302	361	1517	408	1259	728	1374	933	1289	565	811	1601	1199	2307
Std.	25	18	31	17	36	19	33	28	37	26	38	21	26	38	29	45

Table 2. 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}.

As an example, we take $n = 20, \ell = 14$ and simulate 10^3 runs of the combination circuit on random input labels. The resulting number of good labels follows a normal distribution with mean $\mu =\simeq 255.1$ and standard deviation $\sigma =\simeq 14.5$. By integrating the density function on $]-\infty; 200]$ we can estimate the probability to obtain less than 200 labels to be $\leq 2^{-13}$. By doubling the number of labels, the expected m more than doubles: $\mu = 810.2, \sigma = 25.2$, and the same estimate gives a probability of failure lower than 2^{-429}. We remark that the standard deviation obtained is usually small, so when we obtain an average $\mu \geq 700$, we estimate that the probability of failure is negligibly smaller than 2^{-200}. We display some simulation results in Table 2 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

[4] Our code can be found at: `https://project.inria.fr/quasymodo/legendre_quantum_security-tar/`

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.