Legendre Sequences are Pseudorandom under the Quadratic-Residuosity Assumption

Henry Corrigan-Gibbs
MIT

David J. Wu
UT Austin

Abstract. The Legendre sequence of an integer x modulo a prime p with respect to offsets $\vec{a} = (a_1, \ldots, a_\ell)$ is the string of Legendre symbols $(\frac{x+a_1}{p}), \ldots, (\frac{x+a_\ell}{p})$. Under the quadratic-residuosity assumption, we show that the function that maps the pair (x, p) to the Legendre sequence of x modulo p, with respect to public random offsets $\vec{a}$, is a pseudorandom generator. This answers an open question of Damgård (CRYPTO 1988), up to the choice of the offsets $\vec{a}$.

1 Introduction

In a 1988 paper, Damgård [10] proposed a pseudorandom generator (PRG) based on strings of Legendre symbols. In particular, he posited that, for a *secret* random prime p, a secret random value $x \in \mathbb{Z}_p^*$, and any sequence length $\ell = \text{polylog}(p)$, the sequence of Legendre symbols $(\frac{x+1}{p}), (\frac{x+2}{p}), \ldots, (\frac{x+\ell}{p})$ is pseudorandom. Damgård was not able to prove the security of this construction under any pre-existing cryptographic assumption; that problem remains open.

In this work, we show that a generalization of Damgård's PRG is indeed secure under the standard quadratic-residuosity assumption. In particular, we consider a variant of Damgård's generator that outputs the Legendre symbols $(\frac{x+a_1}{p}), (\frac{x+a_2}{p}), \ldots, (\frac{x+a_\ell}{p})$, where the integers $\vec{a} = (a_1, \ldots, a_\ell)$ are part of the generator's public parameters. We prove that, if the public offsets $\vec{a}$ are random values of bitlength $O(\log p)$, the resulting pseudorandom generator is secure under the standard quadratic residuosity assumption.

Along the way, we show that, under the quadratic-residuosity assumption, the function $L((p, x), a) := (\frac{x+a}{p})$, keyed by a secret prime p and offset x, is a weak pseudorandom function. Roughly speaking, no efficient adversary can distinguish the outputs of $L((p, x), \cdot)$ from those of a truly random function, given only access to evaluations of the function at *random* elements of its domain.

Throughout this paper, we focus exclusively on the setting in which the prime modulus p is *secret*—i.e., part of the seed of the pseudorandom generator or the key to the weak pseudorandom function—as in Damgård's original paper. More recent work [2, 3, 13, 16, 18, 22] has studied a variant of Damgård's generator where the prime modulus p is *public*—i.e., part of the public parameters of the pseudorandom generator. Relating the security of the public-prime variant to a more common number-theoretic assumption is a second open problem.

Technical overview. Before sketching our techniques, we recall the quadratic-residuosity assumption. Given an integer $N = pq$, for distinct primes p and q, the quadratic-residuosity problem [5,15] asks an algorithm to determine whether an integer $x \in \mathbb{Z}_N$ with Jacobi symbol 1 modulo N is a quadratic residue modulo N—i.e., whether there exists $y \in \mathbb{Z}_N$ such that $y^2 = x \bmod N$. Throughout, we actually require $p \equiv q \equiv 3 \bmod 4$, which makes $N = pq$ a "Blum integer." We note that restricting the modulus to a Blum integer does not make the quadratic-residuosity problem any easier (we refer to Theorem A.2 in Appendix A for one possible proof of this fact provided to us by Brent Waters [25]).

The proof of our main result proceeds in two steps. Our first step is to show that a new variant of the quadratic-residuosity problem, which we call the *ℓ-time full-domain Legendre problem* (Definition 3.3), is as hard as standard quadratic residuosity. In this new problem, we give an algorithm a Blum integer $N = pq$, a list of independent random values $x_1, \dots, x_\ell \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_N^*$, and bits $(\beta_1, \dots, \beta_\ell) \in \{-1, 1\}^\ell$. The algorithm must distinguish the case in which the β values are the Legendre symbols $(\frac{x_1}{p}), \dots, (\frac{x_\ell}{p})$ modulo one prime factor p of N, or whether the β values are independent random values in $\{-1, 1\}$. The key distinction here from standard quadratic-residuosity problems is that the challenge values $(x_1, \dots, x_\ell)$ are sampled from all of $\mathbb{Z}_N^*$, not just the set of elements with Jacobi symbol 1.

We show that for every $\ell = \mathrm{polylog}(N)$, the ℓ-time full-domain Legendre problem is as hard as standard quadratic residuosity (Corollary 3.6). To do so, we use a standard argument to show that a decisional variant of quadratic residuosity (Definition 3.1) is as hard as standard quadratic residuosity (Lemma 3.5). Via a randomization procedure that exploits the structure of $\mathbb{Z}_N^*$, we then show that hardness of the decisional quadratic-residuosity variant implies that the ℓ-time full-domain-Legendre problem is hard (Lemma 3.2).

The second piece of the proof of our main theorem is to show that distinguishing sequences of Legendre symbols modulo a hidden prime p is as hard as the ℓ-time full-domain-Legendre problem. In particular, our main task is to show that for a secret prime p and secret offset x, for large-enough random integers $a_1, \dots, a_\ell$, distinguishing the string $(\frac{x+a_1}{p}), \dots, (\frac{x+a_\ell}{p})$ from a random string in $\{-1, 1\}^\ell$ is as hard as the ℓ-time full-domain Legendre problem.

In fact, we prove the stronger statement that the function $L((p, x), a) \coloneqq (\frac{x+a}{p})$ is a weak pseudorandom function (PRF) under the ℓ-time full-domain Legendre problem. In this PRF construction, the pair (p, x) is the PRF key and a is the input.

We use an ℓ-query distinguisher for the Legendre weak PRF to construct a distinguisher for the ℓ-time full-domain Legendre problem. Let $(N, x_1, \dots, x_\ell, y_1, \dots, y_\ell)$ be a challenge for the ℓ-time full-domain-Legendre problem. Here, $N = pq$ is a random Blum integer, and for all $i \in [\ell]$, $x_i \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_N^*$ and either $y_i = (\frac{x_i}{p})$ or $y_i \overset{\mathrm{R}}{\leftarrow} \{-1, 1\}$. The reduction algorithm needs to transform this challenge into ℓ evaluations $y'_1, \dots, y'_\ell \in \{-1, 1\}$ of the weak PRF on random points $a_1, \dots, a_\ell$, where either $y'_i = (\frac{x+a_i}{p})$ or $y'_i \overset{\mathrm{R}}{\leftarrow} \{-1, 1\}$, p is a random Blum prime, and x is a random integer. Suppose that p, q are λ-bit Blum primes and

x is a random λ-bit integer. If we sample each a_i to be a random 3λ-bit integer (i.e., $a_i \xleftarrow{\text{R}} [2^{3\lambda}]$), we can rely on a smudging argument to argue that following two distributions are statistically indistinguishable, for *all* choices of x:

$$\left\{x + a_i \bmod p : a_i \xleftarrow{\text{R}} [2^{3\lambda}]\right\} \quad \text{and} \quad \left\{x_i \bmod p : x_i \xleftarrow{\text{R}} \mathbb{Z}_N^*\right\}.$$

In particular, this means the following distributions are also statistically indistinguishable:

$$\left\{\left(\frac{x + a_i}{p}\right) : a_i \xleftarrow{\text{R}} [2^{3\lambda}]\right\} \quad \text{and} \quad \left\{\left(\frac{x_i}{p}\right) : x_i \xleftarrow{\text{R}} \mathbb{Z}_N^*\right\}.$$

This means the reduction algorithm can simply sample $a_i \xleftarrow{\text{R}} [2^{3\lambda}]$ and set $y_i' = y_i$. By the above smudging argument, we can show that when $y_i = (\frac{x_i}{p})$, then the challenge $(a_1, \ldots, a_\ell, y_1', \ldots, y_\ell')$ is statistically close to the pseudorandom distribution for the Legendre weak PRF. If $y_i \xleftarrow{\text{R}} \{-1, 1\}$, then the reduction algorithm perfectly simulates the random distribution. Taken together, this shows that distinguishing the Legendre weak PRF is at least as hard as solving the ℓ-time full-domain-Legendre problem, which in turn is as hard as quadratic residuosity. We conclude then that under the quadratic-residuosity assumption, the Legendre weak PRF is secure. Correspondingly, this implies security of Damgård's PRG with a hidden prime and uniformly random offsets. We give the formal proof in Section 4 (see Lemma 4.3 in Section 4.1 for the analysis of the Legendre weak PRF and Theorem 4.5 in Section 4.2 for the implication to Damgård's PRG).

Related work. We study Legendre-symbol-based cryptographic constructions in which the prime modulus p is *secret*. A more recent conjecture is that these constructions remain secure even when the prime modulus p is *public* [2, 3, 4, 13, 16, 18, 22]. Specifically, these works conjecture that, for a public prime p, on secret key $x \in \mathbb{Z}_p^*$ and input $a \in \mathbb{Z}_p^*$, the function $L_p(x, a) \coloneqq (\frac{x+a}{p})$ is a (strong) pseudorandom function. There are better-than-brute-force attacks against this construction in both the classical [19] and quantum [14] settings. There is also a quantum polynomial-time attack [24], provided that the adversary can make superposition queries to the pseudorandom function. There are no known polynomial-time (or even sub-exponential-time) attacks against the Legendre pseudorandom function in the classical setting.

The public-prime function L_p is of special interest because it has a small arithmetic circuit over $\mathbb{Z}_p$ via the identity $L_p(x, a) = (x + a)^{\frac{p-1}{2}} \bmod p$. The function's arithmetic nature makes it concretely efficient to evaluate inside a multiparty computation [2,4,13,16] or zero-knowledge proof. Some applications, such as to post-quantum signatures [4], only require that the public-prime function L_p is weakly pseudorandom. Because of these many applications, a worthwhile direction for future work is to prove pseudorandomness of the public-prime function L_p under a standard number-theoretic assumption.

Starting from the turn of the 20th century, number theorists have studied the *statistical* properties of Legendre sequences [1,8,11,12,17,20]. Notably, Davenport [11, 12] showed that for any prime p and any sequence of Legendre symbols $\vec{y} \in \{-1, 1\}^{\ell}$, the number of values $x \in \mathbb{Z}_p$ where $\vec{y} = \big((\frac{x+1}{p}), \dots, (\frac{x+\ell}{p})\big)$ is in the range $\frac{p}{2^\ell} \pm \ell \cdot O(p^{\varepsilon})$ for some $\varepsilon \geq \frac{1}{2}$. Davenport's analysis applied to sequences of length $\ell \leq 9$. Using the Weil bound [26], Peralta [20] later improved this bound to show for arbitrary length ℓ, the number of values $x \in \mathbb{Z}_p$ that generates any sequence $\vec{y} \in \{-1, 1\}^{\ell}$ is in the range $\frac{p}{2^\ell} \pm \ell \cdot O(\sqrt{p})$. Both sets of results show that for any fixed length $\ell \in \mathbb{N}$, the distribution of length ℓ (consecutive) Legendre sequences *statistically* converges to the uniform distribution over $\{-1, 1\}^{\ell}$ as p grows. These results provide evidence that Legendre sequences are uniformly distributed, and related heuristics have applications to certain factoring algorithms [21]. In our work, we show that to any *computationally-bounded* adversary, Legendre sequences (with random offsets) are indeed indistinguishable from uniform under the quadratic-residuosity assumption.

In the same 1988 paper that inspired this work [10], Damgård also introduced a pseudorandom generator based on Jacobi symbols. This construction is the same as the one we study, except that it replaces the Legendre symbol modulo p by the Jacobi symbol modulo a composite integer N. More precisely, the Jacobi pseudorandom generator $J_{N,\ell}$ with modulus N, output length ℓ, and seed $x \in \mathbb{Z}_N^*$ outputs the string $J_{N,\ell}(x) := (\frac{x+1}{N}), \dots, (\frac{x+\ell}{N})$. Recent work [9] showed that for a composite modulus of the form $N = p^2 q$, where p, q are distinct primes, the function $J_{N,\ell}$ is one-way assuming the hardness of factoring $N = p^2 q$, together with Boneh and Lipton's number-theoretic conjecture on the uniqueness of residue sequences [6]. Moreover, under an additional stronger conjecture on the uniqueness of Jacobi sequences, they also show that $J_{N,\ell}$ is collision resistant. That work does not show that the function $J_{N,\ell}$ satisfies any sort of pseudorandomness property.

2 Background

Notation. We let $\mathbb{N} = \{1, 2, 3, \dots\}$ denote the natural numbers. We use $x \coloneqq 1$ to denote definition and $x \leftarrow 1$ to denote assignment. Let N be a positive integer. We use $\mathbb{Z}_N^*$ to denote the multiplicative group of integers modulo N. Then, we let $\mathbb{J}_N \subset \mathbb{Z}_N^*$ denote the set of elements of $\mathbb{Z}_N^*$ with Jacobi symbol 1: $\mathbb{J}_N \coloneqq \{x \mid x \in \mathbb{Z}_N^*, \big(\frac{x}{N}\big) = 1\}$. We let $\mathbb{QR}_N \subset \mathbb{J}_N$ denote the set of quadratic residues modulo N: $\mathbb{QR}_N \coloneqq \{x^2 \mid x \in \mathbb{Z}_N^*\}$. For a finite set S, we use $x \xleftarrow{\text{R}} S$ to denote an independent uniform sample from S and we write $\mathsf{Uniform}(S)$ to denote the uniform distribution over S. We say an algorithm is efficient if it runs in probabilistic polynomial time in the length of its input. We say a function $f(\lambda)$ is negligible if $f = o(\lambda^{-c})$ for all $c \in \mathbb{N}$ and denote this by writing $f(\lambda) = \mathsf{negl}(\lambda)$.

We use $\mathsf{BPrimes}_\lambda$ to denote the set of λ-bit primes congruent to 3 mod 4 (i.e., "Blum primes"). A "Blum integer" is the product of two distinct Blum primes.

2.1 The Quadratic-Residuosity Problem

Goldwasser and Micali [15] define the quadratic-residuosity relative to a standard RSA modulus $N = pq$, for arbitrary large (distinct) primes p and q. Later on, Blum, Blum, and Shub [5] introduced a variant that restricts the modulus N to a Blum integer (i.e., $N = pq$ where p, q are primes that are congruent to 3 mod 4).

It is a folklore claim that the Blum-Blum-Shub variant of quadratic residuosity is as hard as the standard Goldwasser-Micali version of the problem. For completeness, we give a self-contained reduction due to Brent Waters [25] in Appendix A that proves this fact. Using the Blum-Blum-Shub variant simplifies our analysis, so we use it throughout since this change does not affect our main results.

Definition 2.1 (Quadratic Residuosity). For an algorithm $\mathcal{A}$, security parameter λ, and bit $b \in \{0,1\}$, we define $\rho_{\mathcal{A},b}(\lambda)$ to be

$$\rho_{\mathcal{A},b}(\lambda) := \Pr\left[\mathcal{A}(N, x_b) = 1 \ : \ \begin{array}{c} p, q \xleftarrow{\mathrm{R}} \mathsf{BPrimes}_\lambda \\ N \leftarrow pq \\ x_0 \xleftarrow{\mathrm{R}} \mathbb{QR}_N, x_1 \xleftarrow{\mathrm{R}} \mathbb{J}_N \setminus \mathbb{QR}_N \end{array}\right].$$

We define the *quadratic-residuosity advantage* of an algorithm $\mathcal{A}$ as:

$$\mathsf{QRAdv}[\mathcal{A}](\lambda) := |\rho_{\mathcal{A},0}(\lambda) - \rho_{\mathcal{A},1}(\lambda)|.$$

2.2 Sampling Quadratic Residues and Non-Residues

Here, we describe how to efficiently sample integers with a given quadratic character. Throughout, we take $N \in \mathbb{N}$ to be a positive integer.

Sampling from $\mathbb{QR}_N$ (quadratic residues). We can efficiently sample $x \xleftarrow{\mathrm{R}} \mathbb{QR}_N$ by sampling $y \xleftarrow{\mathrm{R}} \mathbb{Z}_N^*$ and outputting $x = y^2$.

Blum integers and sampling from $\mathbb{J}_N \setminus \mathbb{QR}_N$ (quadratic non-residues). Recall that a Blum integer is an integer $N = pq$ for distinct primes p and q such that $p \equiv q \equiv 3 \bmod 4$. When N is a Blum integer, -1 is a quadratic non-residue modulo N. Thus, we can sample a random quadratic non-residue modulo N as $-x^2 \bmod N$, where $x \xleftarrow{\mathrm{R}} \mathbb{Z}_N^*$.

Sampling from $\mathbb{J}_N$ (elements of Jacobi symbol 1 or -1). There is an efficient algorithm for computing the Jacobi symbol of an element modulo N without knowing the factorization of N [23, Chapter 12.3]. Since half of the elements of $\mathbb{Z}_N^*$ are in $\mathbb{J}_N$, rejection sampling is enough to sample a random element of either $\mathbb{J}_N$ or $\mathbb{Z}_N^* \setminus \mathbb{J}_N$. After λ samples, such an algorithm fails with probability $2^{-\lambda}$.

2.3 Standard Definitions of Pseudorandomness

Next, we recall some standard notions of pseudorandomness. Since we will be working throughout with Legendre symbols, it will be convenient for us to define standard pseudorandomness notions with respect to values in $\{-1, 1\}$.

Definition 2.2 (Pseudorandom Generator). Let $\mathcal{G} = \{\mathcal{G}_\lambda \mid \lambda \in \mathbb{N}\}$ be a family of sets of functions. For each $\lambda \in \mathbb{N}$, $\mathcal{G}_\lambda$ is a set of efficiently-computable length-increasing functions $G\colon S_G \to \{-1,1\}^{\ell(\lambda)}$, where S_G is a set of seeds, which may differ for each function G. We assume that the function implicitly describes its seed space S_G, and that there is an efficient explicit algorithm for sampling from S_G. Then for an algorithm $\mathcal{A}$, security parameter λ, and bit $b \in \{0,1\}$, define

$$\rho_{\mathcal{A},b}(\lambda) \coloneqq \Pr\left[\mathcal{A}(G,\sigma_b) = 1 \quad : \quad \begin{matrix} G \xleftarrow{\text{R}} \mathcal{G}_\lambda & r \xleftarrow{\text{R}} S_G \\ \sigma_0 \leftarrow G(r) & \sigma_1 \xleftarrow{\text{R}} \{-1,1\}^{\ell(\lambda)} \end{matrix}\right].$$

We define the advantage of algorithm $\mathcal{A}$ at attacking $\mathcal{G}$ as a *pseudorandom generator* as $\mathsf{PRGAdv}[\mathcal{A},\mathcal{G}] \coloneqq |\rho_{\mathcal{A},0}(\lambda) - \rho_{\mathcal{A},1}(\lambda)|$.

Definition 2.3 (Weak Pseudorandom Function). Let $\mathcal{F} = \{F_\lambda \mid \lambda \in \mathbb{N}\}$ be a family of functions, where for all $\lambda \in \mathbb{N}$, F_λ is an efficiently-computable function $F_\lambda\colon K_\lambda \times D_\lambda \to \{-1,1\}$ with key-space K_λ and domain D_λ, respectively. We assume there are efficient explicit algorithms for sampling from K_λ and D_λ. For an algorithm $\mathcal{A}$ and a security parameter λ, we define $\rho_{\mathcal{A},b}(\lambda)$ as

$$\rho_{\mathcal{A},b}(\lambda) \coloneqq \Pr\left[\mathcal{A}^{\mathcal{O}_b}(1^\lambda) = 1 \quad : \quad \begin{matrix} f \xleftarrow{\text{R}} \mathsf{Funs}[D_\lambda, \{-1,1\}] \\ k \xleftarrow{\text{R}} K_\lambda \end{matrix}\right],$$

where $\mathsf{Funs}[D_\lambda, \{-1,1\}]$ is the set of all functions from D_λ to $\{-1,1\}$ and we define the oracles $\mathcal{O}_0$ and $\mathcal{O}_1$ as follows:

- $\mathcal{O}_0$: Sample $x \xleftarrow{\text{R}} D_\lambda$ and output $(x, F_\lambda(k,x))$.
- $\mathcal{O}_1$: Sample $x \xleftarrow{\text{R}} D_\lambda$ and output $(x, f(x))$.

We define the advantage of algorithm $\mathcal{A}$ at attacking $\mathcal{F}$ as a *weak pseudorandom function* as $\mathsf{wPRFAdv}[\mathcal{A},\mathcal{F}](\lambda) \coloneqq |\rho_{\mathcal{A},0}(\lambda) - \rho_{\mathcal{A},1}(\lambda)|$. For a parameter ℓ, we say that $\mathcal{A}$ is an ℓ-query adversary if it makes at most ℓ oracle queries.

3 New Variants of Quadratic Residuosity

This section prepares the ground for the proof of our main result by defining two variants of the quadratic-residuosity problem and showing that both are as hard as the standard version.

3.1 Decisional Quadratic Residuosity is as Hard as Standard Quadratic Residuosity

The following variant of the quadratic residuosity problem asks an algorithm to distinguish, for a Blum integer $N = pq$ and a random element $x \leftarrow \mathbb{Z}_N^*$, the Legendre symbol $(\frac{x}{p})$ from an independent random value in $\{-1,1\}$.

Definition 3.1 (Decisional Quadratic Residuosity). For an algorithm $\mathcal{A}$, security parameter λ, and bit $b \in \{0,1\}$, we define $\rho_{\mathcal{A},b}(\lambda)$ to be

$$\rho_{\mathcal{A},b}(\lambda) \coloneqq \Pr\left[\mathcal{A}(N,x,y_b) = 1 \ : \ \begin{array}{c} p,q \xleftarrow{R} \mathsf{BPrimes}_\lambda, N \leftarrow pq \\ x \xleftarrow{R} \mathbb{J}_N \\ y_0 \leftarrow \left(\frac{x}{p}\right), y_1 \xleftarrow{R} \{-1,1\} \end{array}\right].$$

We define the *decisional-quadratic-residuosity advantage* of an algorithm $\mathcal{A}$ as:

$$\mathsf{QRDAdv}[\mathcal{A}](\lambda) \coloneqq \left|\rho_{\mathcal{A},0}(\lambda) - \rho_{\mathcal{A},1}(\lambda)\right|.$$

We now show that if the quadratic residuosity assumption is hard, then the decisional quadratic residuosity assumption is also hard. This analysis follows readily from Yao's classic argument showing that an unpredictable bit is also pseudorandom [27]. We state the result below:

Lemma 3.2 (If QR is hard, decisional QR is hard). *For every efficient decisional-quadratic-residuosity adversary $\mathcal{A}$, there exists an efficient quadratic-residuosity adversary $\mathcal{B}$ where*

$$\mathsf{QRDAdv}[\mathcal{A}](\lambda) = \frac{1}{2} \cdot \mathsf{QRAdv}[\mathcal{B}](\lambda).$$

Proof. Let $\mathcal{A}$ be an efficient decisional-quadratic-residuosity adversary. We first use Algorithm $\mathcal{A}$ to construct the quadratic-residuosity adversary $\mathcal{B}$ and then we argue about its advantage.

Algorithm $\mathcal{B}$, on input a quadratic-residuosity challenge (N,x), operates as follows:

- Sample $y \xleftarrow{R} \{-1,1\}$.
- Compute $b \leftarrow \mathcal{A}(N,x,y)$. *Without loss of generality, we assume that on all inputs, Algorithm $\mathcal{A}$ outputs a value $b \in \{0,1\}$; any non-conforming algorithm can be transformed into one with this property without decreasing its advantage.*
- If $b = 1$, output y. Otherwise, output $-y$.

To analyze the advantage of Algorithm $\mathcal{B}$, we first define the following quantities, for $z \in \{0,1\}$:

$$\nu_{\mathsf{QR},z}(\lambda) = \Pr\left[\mathcal{A}(N,x,z) = 1 \ : \ p,q \xleftarrow{R} \mathsf{BPrimes}_\lambda, N \leftarrow pq, x \xleftarrow{R} \mathbb{QR}_N\right]$$
$$\nu_{\mathsf{QNR},z}(\lambda) = \Pr\left[\mathcal{A}(N,x,z) = 1 \ : \ p,q \xleftarrow{R} \mathsf{BPrimes}_\lambda, N \leftarrow pq, x \xleftarrow{R} \mathbb{J}_N \setminus \mathbb{QR}_N\right].$$

By definition,

$$\begin{aligned}\mathsf{QRDAdv}[\mathcal{A}](\lambda) &= \Big|\frac{1}{2}(\nu_{\mathsf{QR},1}(\lambda) + \nu_{\mathsf{QNR},-1}(\lambda)) \\ &\qquad - \frac{1}{4}(\nu_{\mathsf{QR},1}(\lambda) + \nu_{\mathsf{QR},-1}(\lambda) + \nu_{\mathsf{QNR},1}(\lambda) + \nu_{\mathsf{QNR},-1}(\lambda))\Big| \\ &= \frac{1}{4}\left|\nu_{\mathsf{QR},1}(\lambda) + \nu_{\mathsf{QNR},-1}(\lambda) - \nu_{\mathsf{QR},-1}(\lambda) - \nu_{\mathsf{QNR},1}(\lambda)\right|.\end{aligned}$$

Let (N, x) be a quadratic-residuosity challenge. We consider the probability that Algorithm $\mathcal{B}$ outputs 1. Algorithm $\mathcal{B}$ only outputs 1 when:

- $\mathcal{A}(N, x, y) = 1$ and $y = 1$, or
- $\mathcal{A}(N, x, y) = -1$ and $y = -1$.

We now compute the probability that Algorithm $\mathcal{B}$ outputs 1 depending on whether $x \in \mathbb{QR}_N$ or $x \in \mathbb{J}_N \setminus \mathbb{QR}_N$. Recall that Algorithm $\mathcal{B}$ samples $y \xleftarrow{\text{R}} \{-1, 1\}$.

- **Case** $x \in \mathbb{QR}_N$**.** Algorithm $\mathcal{B}$ outputs 1 with the following probability:

$$\begin{aligned}\Pr[\mathcal{B} \text{ outputs } 1] &= \underbrace{\frac{1}{2} \cdot \nu_{\mathsf{QR},1}(\lambda)}_{y=1 \text{ and } \mathcal{A}(N,x,y)=1} + \underbrace{\frac{1}{2} \cdot (1 - \nu_{\mathsf{QR},-1}(\lambda))}_{y=-1 \text{ and } \mathcal{A}(N,x,y)=-1} \\ &= \frac{1}{2} + \frac{1}{2}\big(\nu_{\mathsf{QR},1}(\lambda) - \nu_{\mathsf{QR},-1}(\lambda)\big).\end{aligned}$$

- **Case** $x \in \mathbb{J}_N \setminus \mathbb{QR}_N$**.** Algorithm $\mathcal{B}$ outputs 1 with the following probability:

$$\begin{aligned}\Pr\left[\mathcal{B} \text{ outputs } 1\right] &= \underbrace{\frac{1}{2} \cdot \nu_{\mathsf{QNR},1}(\lambda)}_{y=1 \text{ and } \mathcal{A}(N,x,y)=1} + \underbrace{\frac{1}{2} \cdot (1 - \nu_{\mathsf{QNR},-1})}_{y=-1 \text{ and } \mathcal{A}(N,x,y)=-1} \\ &= \frac{1}{2} + \frac{1}{2}\big(\nu_{\mathsf{QNR},1}(\lambda) - \nu_{\mathsf{QNR},-1}(\lambda)\big).\end{aligned}$$

We take the difference of the probability that Algorithm $\mathcal{B}$ outputs 1 in each of the two cases to find the quadratic-residuosity advantage:

$$\begin{aligned}\mathsf{QRAdv}[\mathcal{B}](\lambda) &= \left|\frac{1}{2}\big(\nu_{\mathsf{QR},1}(\lambda) + \nu_{\mathsf{QNR},-1}(\lambda) - \nu_{\mathsf{QR},-1}(\lambda) - \nu_{\mathsf{QNR},1}(\lambda)\big)\right| \\ &= 2 \cdot \mathsf{QRDAdv}[\mathcal{A}](\lambda).\end{aligned}$$ □

3.2 The Full-Domain-Legendre Problem is as Hard as Standard Quadratic Residuosity

Standard quadratic residuosity asks an algorithm to determine, for a modulus $N = pq$, whether a random element of Jacobi symbol 1 modulo N (i.e., in $\mathbb{J}_N$) is a quadratic residue or non-residue modulo N. In the following variant, we ask an algorithm to determine whether a random element of the full group $\mathbb{Z}_N^*$ is a quadratic residue or non-residue modulo a prime factor p of N. The key difference is that here, we sample the challenge element from all of $\mathbb{Z}_N^*$, rather than the smaller set of elements $\mathbb{J}_N$.

This subsection studies the following computational problem:

Definition 3.3 (ℓ-Time Full-Domain Legendre Problem). For an algorithm $\mathcal{A}$, security parameter λ, challenge length ℓ, and a bit $b \in \{0, 1\}$, we define

$\rho_{\mathcal{A},b}(\lambda)$ to be

$$\rho_{\mathcal{A},b}(\lambda) \coloneqq \Pr\left[\mathcal{A}\left(N, \begin{matrix} x_1,\dots,x_\ell \\ y_1^{(b)},\dots,y_\ell^{(b)} \end{matrix}\right) = 1 : \begin{matrix} p,q \xleftarrow{\mathrm{R}} \mathsf{BPrimes}_\lambda, N \leftarrow pq \\ x_1,\dots,x_\ell \xleftarrow{\mathrm{R}} \mathbb{Z}_N^* \\ \forall i \in [\ell] : y_i^{(0)} \leftarrow \left(\frac{x_i}{p}\right) \\ \forall i \in [\ell] : y_i^{(1)} \xleftarrow{\mathrm{R}} \{-1,1\} \end{matrix}\right].$$

We define the *ℓ-time full-domain-Legendre advantage* of $\mathcal{A}$ to be

$$\mathsf{LegAdv}_\ell[\mathcal{A}](\lambda) \coloneqq |\rho_{\mathcal{A},0}(\lambda) - \rho_{\mathcal{A},1}(\lambda)|.$$

The following lemma is key to the proof of Definition 3.3:

Lemma 3.4. *For all composite integers $N = pq$ (where p, q are distinct primes) and for every*

- *$z_0 \in \mathbb{Z}_N^*$ that is a quadratic non-residue modulo both p and q, and*
- *$z_1 \in \mathbb{Z}_N^*$ that is a quadratic non-residue modulo p and a residue modulo q,*

and every value $u \in \mathbb{J}_N$, the following distributions are identical:

$$\left\{(x,\beta) : \begin{matrix} r \xleftarrow{\mathrm{R}} \mathbb{QR}_N, \quad \rho_0,\rho_1 \xleftarrow{\mathrm{R}} \{0,1\} \\ x \leftarrow urz_0^{\rho_0} z_1^{\rho_1} \in \mathbb{Z}_N^* \\ \beta \leftarrow (-1)^{\rho_0+\rho_1}\left(\frac{u}{p}\right) \end{matrix}\right\} \equiv \left\{(x,\beta) : \begin{matrix} x \xleftarrow{\mathrm{R}} \mathbb{Z}_N^* \\ \beta \leftarrow \left(\frac{x}{p}\right) \end{matrix}\right\}.$$

Proof. We prove the following claim before returning to the proof of the lemma.

Claim. *For all N, z_0, z_1 as in Lemma 3.4 and for all $u \in \mathbb{J}_N$, the first component $x \in \mathbb{Z}_N^*$ of the left-side distribution of Lemma 3.4 is distributed uniformly over $\mathbb{Z}_N^*$.*

Proof of Claim. We prove the lemma by showing that $rz_0^{\rho_0} z_1^{\rho_1}$, as computed in the lemma, is distributed uniformly over $\mathbb{Z}_N^*$.

To do so, it is sufficient to show two things about the mapping $\mu_{z_0,z_1} : \mathbb{QR}_N \times \{0,1\}^2 \to \mathbb{Z}_N^*$ that takes $(r,\rho_0,\rho_1) \mapsto rz_0^{\rho_0} z_1^{\rho_1}$: (1) that the domain and co-domain of have equal size; and (2) that it is invertible.

For the first part: The domain has size $|\mathbb{QR}_N| \cdot 4 = 4 \cdot (\phi(N)/4) = \phi(N)$, where $\phi(\cdot)$ is Euler's totient function. The co-domain has size $|\mathbb{Z}_N^*| = \phi(N)$.

For the second part, we show that the mapping μ_{z_0,z_1} is injective. Take any pair (r,ρ_0,ρ_1) and (r',ρ_0',ρ_1') where $\mu_{z_0,z_1}(r,\rho_0,\rho_1) = \mu_{z_0,z_1}(r',\rho_0',\rho_1')$. By definition of μ_{z_0,z_1}, this means

$$(r/r') z_0^{\rho_0-\rho_0'} z_1^{\rho_1-\rho_1'} = 1 \in \mathbb{Z}_N^*. \tag{1}$$

This means that

$$\left(\frac{r/r'}{p}\right)\left(\frac{z_0^{\rho_0-\rho_0'}}{p}\right)\left(\frac{z_1^{\rho_1-\rho_1'}}{p}\right) = 1 = \left(\frac{r/r'}{q}\right)\left(\frac{z_0^{\rho_0-\rho_0'}}{q}\right)\left(\frac{z_1^{\rho_1-\rho_1'}}{q}\right). \tag{2}$$

Since $r, r' \in \mathbb{QR}_N$, z_0 is a quadratic non-residue modulo q and z_1 is a quadratic residue modulo q, Eq. (2) modulo q asserts that

$$1 = (-1)^{\rho_0 - \rho_0'}.$$

Since $\rho_0, \rho_0' \in \{0, 1\}$, this means that $\rho_0 = \rho_0'$. Next, z_0, z_1 are both quadratic non-residues modulo p, so Eq. (2) modulo p asserts that

$$1 = (-1)^{\rho_0 - \rho_0'} \cdot (-1)^{\rho_1 - \rho_1'} = (-1)^{\rho_1 - \rho_1'}.$$

Since $\rho_1, \rho_1' \in \{0, 1\}$, this also means that $\rho_1 = \rho_1'$. Since $\rho_0 = \rho_0'$ and $\rho_1 = \rho_1'$, Eq. (1) now implies that $r = r'$, in which case we have $(r, \rho_0, \rho_1) = (r', \rho_0', \rho_1')$, as required. The claim follows. □

All that remains is to argue about the second component β in the lemma's left-hand distribution is equal to $\left(\frac{x}{p}\right)$. This holds via the multiplicative nature of the Jacobi symbol:

$$\left(\frac{x}{p}\right) = \left(\frac{u}{p}\right)\left(\frac{r}{p}\right)\left(\frac{z_0}{p}\right)^{\rho_0}\left(\frac{z_1}{p}\right)^{\rho_1} = \alpha \cdot 1 \cdot (-1)^{\rho_0} \cdot (-1)^{\rho_1} = \beta,$$

since $r \in \mathbb{QR}_N$ and z_0, z_1 are non-residues modulo p. □

Lemma 3.5 (If decisional QR is hard, ℓ-time full-domain Legendre is hard). *For every polynomial $\ell = \ell(\lambda)$ and every efficient ℓ-time full-domain-Legendre adversary $\mathcal{A}$, there exists an efficient decisional-Legendre adversary $\mathcal{B}$ where*

$$\mathsf{LegAdv}_\ell[\mathcal{A}](\lambda) = \ell \cdot \mathsf{QRDAdv}[\mathcal{B}](\lambda).$$

Proof. Take any polynomial $\ell = \ell(\lambda)$ and any efficient adversary $\mathcal{A}$ for the ℓ-time full-domain Legendre problem. For each $i \in [0, \ell]$, we define a hybrid experiment Hyb_i that is (implicitly) parameterized by a security parameter λ:

- Hyb_i: In this experiment, the challenger starts by sampling $p, q \xleftarrow{R} \mathsf{BPrimes}_\lambda$ and sets $N \leftarrow pq$. Then, it samples $x_1, \ldots, x_\ell \xleftarrow{R} \mathbb{Z}_N^*$. For all $j \in [i]$, the challenger computes $\beta_j \xleftarrow{R} \{-1, 1\}$ and for all $j \in [i+1, \ell]$, the challenger computes $\beta_j \leftarrow \left(\frac{x_i}{p}\right)$. It gives the tuple $(N, (x_1, \beta_1), \ldots, (x_\ell, \beta_\ell))$ to $\mathcal{A}$. The output of the experiment is whatever $\mathcal{A}$ outputs.

For all $i \in \{0, \ldots, \ell\}$, we write $\mathsf{Hyb}_i(\mathcal{A})$ to denote the output distribution of an execution of Hyb_i with adversary $\mathcal{A}$ (and an implicit security parameter λ). For convenience, we define the following function:

The algorithm is parameterized by an integer $N \in \mathbb{N}$ and values $z_0, z_1 \in \mathbb{Z}_N^*$.

```
Rerand_{N,z_0,z_1}($u \in \mathbb{Z}_N^*,\ y \in \{-1, 1\}$) :
  – Sample $r \xleftarrow{R} \mathbb{QR}_N$ and $\rho_0, \rho_1 \xleftarrow{R} \{-1, 1\}$.
  – Let $x \leftarrow u r z_0^{\rho_0} z_1^{\rho_1} \in \mathbb{Z}_N^*$   and   $\beta \leftarrow (-1)^{\rho_0+\rho_1} \cdot y \in \{-1, 1\}$.
  – Return $(x, \beta)$.
```

By definition of the hybrids,

$$\mathsf{LegAdv}_\ell[\mathcal{A}](\lambda) = |\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_\ell(\mathcal{A}) = 1]|$$

We now use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ where

$$\mathsf{QRDAdv}[\mathcal{B}](\lambda) = \mathsf{LegAdv}_\ell[\mathcal{A}](\lambda) = |\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_\ell(\mathcal{A}) = 1]|.$$

Algorithm $\mathcal{B}$ works as follows:

– Algorithm $\mathcal{B}$ receives a decisional-quadratic-residuosity challenge (N, x, y) and samples an index $i^* \xleftarrow{R} [\ell]$.
– Algorithm $\mathcal{B}$ fixes elements $z_0 \in \mathbb{Z}_N^* \setminus \mathbb{QR}_N$ and $z_1 \in \mathbb{Z}_N^* \setminus \mathbb{J}_N$, using the sampling techniques described in Section 2.2.
– For each $j \in [\ell]$, Algorithm $\mathcal{B}$ now constructs (x_j, β_j) as follows:
  - If $j < i^*$, sample $x_j \xleftarrow{R} \mathbb{Z}_N^*$ and $\beta_j \xleftarrow{R} \{-1, 1\}$.
  - If $j = i^*$, set $(x_j, \beta_j) \xleftarrow{R} \mathsf{Rerand}_{N,z_0,z_1}(x, y)$.
  - If $j > i^*$, set $(x_j, \beta_j) \leftarrow \mathsf{Rerand}_{N,z_0,z_1}(1, 1)$.
– Algorithm $\mathcal{B}$ invokes $\mathcal{A}$ on input $(N, (x_1, \beta_1), \dots, (x_\ell, \beta_\ell))$, and outputs whatever $\mathcal{A}$ outputs.

In the following analysis, we assume that z_1 is a quadratic non-residue modulo p and a quadratic residue modulo q. The reverse case where z_1 is a quadratic non-residue modulo q and a quadratic residue modulo p follows by an analogous argument, where we interchange the roles of p and q. Note that the roles of p, q are symmetric because they are identically distributed (i.e., $p, q \xleftarrow{R} \mathsf{BPrimes}_\lambda$), and moreover, for all $x \in \mathbb{J}_N$, $(\frac{x}{p}) = (\frac{x}{q})$.

We now compute the advantage of $\mathcal{B}$ conditioned on a particular value of $i^* = i$. First, we analyze the distribution of values (x_j, β_j) for all $j \in [\ell] \setminus \{i\}$.

– If $j < i$, the values (x_j, β_j) are independent random values in $\mathbb{Z}_N^* \times \{-1, 1\}$, just as in Hyb_{i-1} and Hyb_i.
– If $j > i$, by Lemma 3.4, the values (x_j, β_j) are distributed exactly as $\big(x_j, (\frac{x_j}{p})\big)$ where $x_j \xleftarrow{R} \mathbb{Z}_N^*$, just as in Hyb_{i-1} and Hyb_i.

What remains is to analyze the distribution of the pair (x_i, β_i). The decisional-quadratic-residuosity challenger samples $p, q \xleftarrow{R} \mathsf{BPrimes}_\lambda$ and sets $N = pq$. It also samples $x \xleftarrow{R} \mathbb{J}_N$. We consider the two possibilities for (x, y):

– By Lemma 3.4, if $y = (\frac{x}{p})$, then (x_i, β_i) is distributed exactly as $\big(x_i, (\frac{x_i}{p})\big)$ for $x_i \xleftarrow{R} \mathbb{Z}_N^*$. Therefore, the distribution of values $(N, x_1, \dots, x_\ell, \beta_1, \dots, \beta_\ell)$ is exactly as in Hyb_{i-1}. The output of $\mathcal{B}$ is then distributed exactly according to the output distribution of Hyb_{i-1} for $\mathcal{A}$. Thus, when $i^* = i$, Algorithm $\mathcal{B}$ outputs 1 with probability $\Pr[\mathsf{Hyb}_{i-1}(\mathcal{A}) = 1]$.
– If $y \xleftarrow{R} \{-1, 1\}$, then, again by Lemma 3.4, the value x_i is distributed exactly as $x_i \xleftarrow{R} \mathbb{Z}_N^*$. In addition, since $\beta_i = (-1)^{\rho_0 + \rho_1} y$, for a value $y \xleftarrow{R} \{-1, 1\}$ that is independent of ρ_0 and ρ_1, the value β_i is distributed as $\beta_i \xleftarrow{R} \{-1, 1\}$. Then the values $(N, x_1, \dots, x_\ell, \beta_1, \dots, \beta_\ell)$ are distributed exactly as in Hyb_i. The output of $\mathcal{B}$ is distributed exactly according to the output distribution

of Hyb_i for $\mathcal{A}$. Thus, when $i^* = i$, Algorithm $\mathcal{B}$ outputs 1 with probability $\Pr[\mathsf{Hyb}_i(\mathcal{A}) = 1]$.

Since $\mathcal{B}$ samples $i^* \xleftarrow{\text{R}} [\ell]$, our analysis above shows that

$$\rho_{\mathcal{B},0}(\lambda) = \frac{1}{\ell} \sum_{i \in [\ell]} \Pr[\mathsf{Hyb}_{i-1}(\mathcal{A}) = 1] \quad \text{and} \quad \rho_{\mathcal{B},1}(\lambda) = \frac{1}{\ell} \sum_{i \in [\ell]} \Pr[\mathsf{Hyb}_i(\mathcal{A}) = 1],$$

where $\rho_{\mathcal{B},b}(\lambda)$ is the function from Definition 3.1. The advantage of Algorithm $\mathcal{B}$ is then

$$\begin{aligned} \mathsf{QRDAdv}[\mathcal{B}](\lambda) &= |\rho_{\mathcal{B},0}(\lambda) - \rho_{\mathcal{B},1}(\lambda)| \\ &= \frac{1}{\ell} |\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_\ell(\mathcal{A}) = 1]| \\ &= \frac{1}{\ell} \mathsf{LegAdv}_\ell[\mathcal{A}](\lambda). \end{aligned}$$

We conclude that

$$\mathsf{LegAdv}_\ell[\mathcal{A}](\lambda) = \ell \cdot \mathsf{QRDAdv}[\mathcal{B}](\lambda). \qquad \square$$

Corollary 3.6 (If QR is hard, full-domain Legendre is hard). *For every polynomial $\ell = \ell(\lambda)$ and every efficient adversary $\mathcal{A}$ with ℓ-time full-domain-Legendre advantage $\mathsf{LegAdv}_\ell[\mathcal{A}](\lambda)$, there exists an efficient adversary $\mathcal{B}$ with quadratic-residuosity advantage $\mathsf{QRAdv}[\mathcal{B}](\lambda)$ where*

$$\mathsf{LegAdv}_\ell[\mathcal{A}](\lambda) = \ell \cdot \mathsf{QRAdv}[\mathcal{B}](\lambda).$$

Proof. An immediate consequence of Lemmas 3.2 and 3.5. $\square$

4 Secret-Prime Legendre Sequences are Pseudorandom Under the Quadratic-Residuosity Assumption

In this section, we show that Legendre sequences with a secret prime and random offsets are pseudorandom, assuming the quadratic-residuosity assumption. This answers the original question of Damgård [10], up to the choice of offsets. We will first show (Section 4.1) the *stronger* statement that a simple generalization of Damgård's pseudorandom generator is actually a weak pseudorandom function. We obtain the Legendre pseudorandom generator, by fixing the evaluation points of the weak pseudorandom function in advance. We then (Section 4.2) show that the Legendre pseudorandom generator is secure under quadratic residuosity.

4.1 Analysis of the Legendre Weak Pseudorandom Function

This subsection studies the following construction:

Definition 4.1 (Hidden-Prime Legendre Weak PRF). Let λ be a security parameter. Define the function $L_\lambda \colon (\mathsf{BPrimes}_\lambda \times [2^\lambda]) \times [2^{3\lambda}] \to \{-1, 1\}$ where

$$L_\lambda((p, x), a) := \left(\frac{x+a}{p}\right).$$

We define $\mathcal{L}^{\mathsf{wPRF}} = \{L_\lambda\}_{\lambda \in \mathbb{N}}$ to be the hidden-prime Legendre weak pseudorandom function.

Much prior work has conjectured that the public-prime variant of the Legendre PRF [2,3,13,16,18,22] satisfies *strong* pseudorandomness, where the adversary is able to adaptively choose the inputs to the oracle in the pseudorandomness game. For some of these applications (e.g., [4]), a weak PRF also suffices.

We prove that the hidden-prime Legendre weak PRF Definition 4.1 satisfies weak pseudorandomness under quadratic residuosity:

Theorem 4.2 (Hidden-Prime Legendre Weak PRF is Secure Under Quadratic Residuosity). *For every efficient ℓ-query algorithm $\mathcal{A}$ that breaks weak pseudorandomness of $\mathcal{L}^{\mathsf{wPRF}}$ with advantage $\mathsf{wPRFAdv}[\mathcal{A}](\lambda)$, there exists an efficient algorithm $\mathcal{B}$ for the quadratic-residuosity problem with advantage $\mathsf{QRAdv}_\ell[\mathcal{B}](\lambda)$ and a negligible function $\mathsf{negl}(\cdot)$ such that*

$$\mathsf{wPRFAdv}[\mathcal{A}, \mathcal{L}^{\mathsf{wPRF}}](\lambda) = \ell \cdot \mathsf{QRAdv}[\mathcal{B}](\lambda) + \mathsf{negl}(\lambda).$$

Proof idea. We prove Theorem 4.2 by showing that any ℓ-query adversary that can break pseudorandomness of the hidden-prime Legendre weak PRF implies an adversary that can break the ℓ-time full-domain-Legendre problem (Definition 3.3). We rely on a smudging argument. To illustrate, consider the following two distributions:

– **Legendre weak PRF outputs:** In this distribution, an adversary making exactly ℓ queries sees a random vector $\vec{a} = (a_1, \ldots, a_\ell) \xleftarrow{\text{R}} [2^{3\lambda}]^\ell$ and Legendre symbols $(\frac{x+a_1}{p}), \ldots, (\frac{x+a_\ell}{p})$, where $p \xleftarrow{\text{R}} \mathsf{BPrimes}_\lambda$ and $x \xleftarrow{\text{R}} [2^\lambda]$.

– **ℓ-time full-domain Legendre instance:** In this distribution, the adversary sees a random vector $\vec{x} = (x_1, \ldots, x_\ell) \xleftarrow{\text{R}} \mathbb{Z}_N^*$ and Legendre symbols $(\frac{x_1}{p}), \ldots, (\frac{x_\ell}{p})$, where $p, q \xleftarrow{\text{R}} \mathsf{BPrimes}_\lambda$, $N = pq$, and $x_1, \ldots, x_\ell \xleftarrow{\text{R}} \mathbb{Z}_N^*$.

The key argument in the proof is that when $a_i \xleftarrow{\text{R}} [2^{3\lambda}]$, the distribution of $(a_i + x \bmod p)$ is statistically indistinguishable from $(x_i \bmod p)$ where $x_i \xleftarrow{\text{R}} \mathbb{Z}_N^*$. In this case, the Legendre symbols from the ℓ-time full-domain Legendre instance can be used to simulate the weak PRF outputs. Since the Legendre symbols in the ℓ-time full-domain Legendre instance is computationally indistinguishable from uniform random, the same applies to the weak PRF outputs and the claim holds. We now give the proof.

Proof of Theorem 4.2. The key lemma required to prove Theorem 4.2 is the following:

Lemma 4.3. *For every polynomial $\ell = \ell(\lambda)$, every efficient ℓ-query algorithm $\mathcal{A}$ that breaks the pseudorandomness of $\mathcal{L}^{\mathsf{wPRF}}$ with advantage $\mathsf{wPRFAdv}[\mathcal{A}](\lambda)$, there exists an efficient algorithm $\mathcal{B}$ for the ℓ-time full-domain Legendre problem with advantage $\mathsf{LegAdv}_\ell[\mathcal{B}](\lambda)$ and a negligible function $\mathsf{negl}(\cdot)$ where*

$$\mathsf{wPRFAdv}[\mathcal{A}, \mathcal{L}^{\mathsf{wPRF}}] = \mathsf{LegAdv}_\ell[\mathcal{B}](\lambda) + \mathsf{negl}(\lambda).$$

Proof. Take any polynomial $\ell = \ell(\lambda)$ and let $\mathcal{A}$ be an efficient ℓ-query algorithm for the weak PRF distinguishing game. Without loss of generality, assume that $\mathcal{A}$ makes exactly ℓ queries. Any adversary that makes fewer than ℓ queries can be generically transformed into one that makes exactly ℓ queries with no loss in advantage. We begin by defining a sequence of hybrid experiments:

- Hyb_0: This is the "pseudorandom" experiment. Namely, the challenger starts by sampling the following components:
  - $p \xleftarrow{R} \mathsf{BPrimes}_\lambda$ and $x \xleftarrow{R} [2^\lambda]$;
  - $a_i \xleftarrow{R} [2^{3\lambda}]$ for all $i \in [\ell]$;
  - $\vec{a} \leftarrow (a_1, \ldots, a_\ell)$ and $\vec{y} \leftarrow \left[\left(\frac{x+a_1}{p}\right), \left(\frac{x+a_2}{p}\right), \ldots, \left(\frac{x+a_\ell}{p}\right)\right]$.

  The challenger gives $(\vec{a}, \vec{y})$ to Algorithm $\mathcal{A}$ as the answer to its ℓ queries. Algorithm $\mathcal{A}$ outputs a bit $b \in \{0,1\}$, which is the output of the experiment.
- Hyb_1: Same as Hyb_0 except the challenger changes how it samples a_i:
  - $p \xleftarrow{R} \mathsf{BPrimes}_\lambda$ and $x \xleftarrow{R} [2^\lambda]$;
  - $a'_i \xleftarrow{R} [2^{3\lambda}]$ and $a_i \leftarrow a'_i - x$ for all $i \in [\ell]$;
  - $\vec{a} \leftarrow (a_1, \ldots, a_\ell)$ and $\vec{y} \leftarrow \left[\left(\frac{a'_1}{p}\right), \left(\frac{a'_2}{p}\right), \ldots, \left(\frac{a'_\ell}{p}\right)\right]$.
- Hyb_2: Same as Hyb_1 except the challenger changes how it samples a'_i:
  - $p, q \xleftarrow{R} \mathsf{BPrimes}_\lambda$, $N \leftarrow pq$, and $x \xleftarrow{R} [2^\lambda]$;
  - $B \leftarrow \lfloor 2^{3\lambda}/N \rfloor$;
  - $x_i \xleftarrow{R} \mathbb{Z}_N$, $w_i \xleftarrow{R} [0, B-1]$, $a'_i \leftarrow w_i N + x_i$ (computed over the *integers*), and $a_i \leftarrow a'_i - x$ for all $i \in [\ell]$;
  - $\vec{a} \leftarrow (a_1, \ldots, a_\ell)$ and $\vec{y} \leftarrow \left[\left(\frac{x_1}{p}\right), \left(\frac{x_2}{p}\right), \ldots, \left(\frac{x_\ell}{p}\right)\right]$.
- Hyb_3: Same as Hyb_2 except the challenger samples $x_i \xleftarrow{R} \mathbb{Z}_N^*$:
  - $p, q \xleftarrow{R} \mathsf{BPrimes}_\lambda$, $N \leftarrow pq$, and $x \xleftarrow{R} [2^\lambda]$;
  - $B \leftarrow \lfloor 2^{3\lambda}/N \rfloor$;
  - $x_i \xleftarrow{R} \mathbb{Z}_N^*$, $w_i \xleftarrow{R} [0, B-1]$, $a'_i \leftarrow w_i N + x_i$ (computed over the *integers*), and $a_i \leftarrow a'_i - x$ for all $i \in [\ell]$;
  - $\vec{a} \leftarrow (a_1, \ldots, a_\ell)$ and $\vec{y} \leftarrow \left[\left(\frac{x_1}{p}\right), \left(\frac{x_2}{p}\right), \ldots, \left(\frac{x_\ell}{p}\right)\right]$.
- Hyb_4: Same as Hyb_3 except the challenger samples $\vec{y} \xleftarrow{R} \{-1,1\}^\ell$:
  - $p, q \xleftarrow{R} \mathsf{BPrimes}_\lambda$, $N \leftarrow pq$, and $x \xleftarrow{R} [2^\lambda]$;
  - $B \leftarrow \lfloor 2^{3\lambda}/N \rfloor$;

  - $x_i \xleftarrow{R} \mathbb{Z}_N^*$, $w_i \xleftarrow{R} [0, B-1]$, $a'_i \leftarrow w_i N + x_i$ (computed over the *integers*), and $a_i \leftarrow a'_i - x$ for all $i \in [\ell]$;
  - $\vec{a} \leftarrow (a_1, \ldots, a_\ell)$ and $\vec{y} \xleftarrow{R} \{-1, 1\}^\ell$.

– Hyb_5: Same as Hyb_4 except the challenger samples $x_i \xleftarrow{R} \mathbb{Z}_N$:
  - $p, q \xleftarrow{R} \mathsf{BPrimes}_\lambda$, $N \leftarrow pq$, and $x \xleftarrow{R} [2^\lambda]$;
  - $B \leftarrow \lfloor 2^{3\lambda}/N \rfloor$;
  - $x_i \xleftarrow{R} \mathbb{Z}_N$, $w_i \xleftarrow{R} [0, B-1]$, $a'_i \leftarrow w_i N + x_i$ (computed over the *integers*), and $a_i \leftarrow a'_i - x$ for all $i \in [\ell]$;
  - $\vec{a} \leftarrow (a_1, \ldots, a_\ell)$ and $\vec{y} \xleftarrow{R} \{-1, 1\}^\ell$.

– Hyb_6: Same as Hyb_5 except the challenger changes how it samples a'_i:
  - $p \xleftarrow{R} \mathsf{BPrimes}_\lambda$ and $x \xleftarrow{R} [2^\lambda]$;
  - $a'_i \xleftarrow{R} [2^{3\lambda}]$ and $a_i \leftarrow a'_i - x$ for all $i \in [\ell]$;
  - $\vec{a} \leftarrow (a_1, \ldots, a_\ell)$ and $\vec{y} \xleftarrow{R} \{-1, 1\}^\ell$.

– Hyb_7: Same as Hyb_4 except the challenger changes how it samples a_i:
  - $p \xleftarrow{R} \mathsf{BPrimes}_\lambda$ and $x \xleftarrow{R} [2^\lambda]$;
  - $a_i \xleftarrow{R} [2^{3\lambda}]$ for all $i \in [\ell]$;
  - $\vec{a} \leftarrow (a_1, \ldots, a_\ell)$ and $\vec{y} \xleftarrow{R} \{-1, 1\}^\ell$.

  This is the "random" experiment.

We write $\mathsf{Hyb}_i(\mathcal{A})$ to denote an output of hybrid Hyb_i with Algorithm $\mathcal{A}$. We now show that the outputs of each pair of adjacent experiments are indistinguishable:

– $\mathsf{Hyb}_0(\mathcal{A})$ and $\mathsf{Hyb}_1(\mathcal{A})$ are statistically indistinguishable. First, consider the distribution of a_i in the two experiments:
  - In Hyb_0, $a_i \xleftarrow{R} [2^{3\lambda}]$.
  - In Hyb_1, $a_i \leftarrow a'_i - x$, where $a'_i \xleftarrow{R} [2^{3\lambda}]$ and $x \in [2^\lambda]$.

  The statistical distance between these two distributions is $2^\lambda/2^{3\lambda} = \mathsf{negl}(\lambda)$. Finally, the vector $\vec{y}$ in Hyb_1 is constructed exactly as in Hyb_0: namely, in Hyb_1, we have $x + a_i = x + a'_i - x = a'_i$. The claim then follows by a hybrid argument over each a_i for $i \in [\ell]$ and the fact that $\ell = \ell(\lambda)$ is polynomially-bounded.

– $\mathsf{Hyb}_1(\mathcal{A})$ and $\mathsf{Hyb}_2(\mathcal{A})$ are statistically indistinguishable. First, consider the distribution of a'_i in the two experiments:
  - In Hyb_1, $a'_i \xleftarrow{R} [2^{3\lambda}]$.
  - In Hyb_2, $a'_i = w_i N + x_i$ where $w_i \xleftarrow{R} [0, B-1]$ and $x_i \xleftarrow{R} \mathbb{Z}_N$.

  By construction, the distribution of a'_i in Hyb_2 is $\mathsf{Uniform}([0, NB-1])$. In Hyb_2, we have $B = \lfloor 2^{3\lambda}/N \rfloor$. This means $2^{3\lambda} = NB + r$ where $r \in [0, N-1]$. Since $N < 2^{2\lambda}$, the statistical distance between $\mathsf{Uniform}([2^{3\lambda}])$ and $\mathsf{Uniform}([0, NB-1])$ is bounded by $r/2^{3\lambda} < N/2^{3\lambda} < 2^{-\lambda} = \mathsf{negl}(\lambda)$. Finally, the vector $\vec{y}$ in Hyb_2 is constructed exactly as in Hyb_1: namely, in

Hyb_2, we have for all $i \in [\ell]$,

$$\left(\frac{a'_i}{p}\right) = \left(\frac{w_i N + x_i}{p}\right) = \left(\frac{w_i N + x_i \bmod p}{p}\right) = \left(\frac{x_i}{p}\right).$$

The claim then follows by a hybrid argument over each a_i for $i \in [\ell]$.

– $\mathsf{Hyb}_2(\mathcal{A})$ and $\mathsf{Hyb}_3(\mathcal{A})$ are statistically indistinguishable. The only difference between these two distributions is the distribution of x_i. In Hyb_2, each x_i is uniform over $\mathbb{Z}_N$ while in Hyb_3, each x_i is uniform over $\mathbb{Z}_N^*$. Since $N = pq$ is a product of two λ-bit primes, the statistical distance between these two distributions is $(p+q-1)/N = \mathsf{negl}(\lambda)$. The claim again follows by a hybrid argument over each x_i for $i \in [\ell]$.

– We show that there exists an efficient adversary $\mathcal{B}$ for the ℓ-time full-domain Legendre problem where

$$\mathsf{LegAdv}_\ell[\mathcal{B}](\lambda) = |\Pr[\mathsf{Hyb}_3(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_4(\mathcal{A}) = 1]|. \tag{3}$$

Algorithm $\mathcal{B}$ works as follows:

- On input a modulus N and a challenge $(x_1, y_1), \ldots, (x_\ell, y_\ell)$, Algorithm $\mathcal{B}$ samples $x \xleftarrow{\text{R}} [2^\lambda]$ and for each $i \in [\ell]$, it samples $w_i \xleftarrow{\text{R}} [0, B-1]$, where $B = \lfloor 2^{3\lambda}/N \rfloor$. It then sets $a'_i \leftarrow w_i N + x_i$ and $a_i \leftarrow a'_i - x$ for all $i \in [\ell]$.
- Algorithm $\mathcal{B}$ gives $\vec{a} \leftarrow (a_1, \ldots, a_\ell)$ and $\vec{y} \leftarrow (y_1, \ldots, y_\ell)$ to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs.

By definition, the ℓ-time full-domain Legendre challenger samples $p, q \xleftarrow{\text{R}} \mathsf{BPrimes}_\lambda$, sets $N = pq$, and samples $x_1, \ldots, x_N \xleftarrow{\text{R}} \mathbb{Z}_N^*$. When $y_i = (\frac{x_i}{p})$, then Algorithm $\mathcal{B}$ perfectly simulates an execution of Hyb_3 and outputs 1 with probability $\Pr[\mathsf{Hyb}_3(\mathcal{A}) = 1]$. When $y_i \xleftarrow{\text{R}} \{-1, 1\}$, then Algorithm $\mathcal{B}$ perfectly simulates an execution of Hyb_4 and outputs 1 with probability $\Pr[\mathsf{Hyb}_4(\mathcal{A}) = 1]$. Thus, Eq. (3) holds.

– $\mathsf{Hyb}_4(\mathcal{A})$ and $\mathsf{Hyb}_5(\mathcal{A})$ are statistically indistinguishable by the same argument as used to argue statistical indistinguishability of $\mathsf{Hyb}_2(\mathcal{A})$ and $\mathsf{Hyb}_3(\mathcal{A})$.

– $\mathsf{Hyb}_5(\mathcal{A})$ and $\mathsf{Hyb}_6(\mathcal{A})$ are statistically indistinguishable by the same argument as used to argue statistical indistinguishability of $\mathsf{Hyb}_1(\mathcal{A})$ and $\mathsf{Hyb}_2(\mathcal{A})$.

– $\mathsf{Hyb}_6(\mathcal{A})$ and $\mathsf{Hyb}_7(\mathcal{A})$ are statistically indistinguishable be the same argument as used to argue statistical indistinguishability of $\mathsf{Hyb}_0(\mathcal{A})$ and $\mathsf{Hyb}_1(\mathcal{A})$.

By a hybrid argument, we conclude that there exists a negligible function $\mathsf{negl}(\cdot)$ and an efficient adversary $\mathcal{B}$ such that for all $\lambda \in \mathbb{N}$,

$$\begin{aligned}\mathsf{PRGAdv}[\mathcal{A}, \mathcal{L}_\ell^{\mathsf{PRG}}](\lambda) &= |\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_7(\mathcal{A}) = 1]| \\ &= \mathsf{LegAdv}_\ell[\mathcal{B}](\lambda) + \mathsf{negl}(\lambda).\end{aligned}$$ □

Theorem 4.2 then follows from the fact that breaking the hidden-prime Legendre weak PRF is as hard as the ℓ-time full-domain Legendre problem (Lemma 4.3) and the fact that the ℓ-time full-domain Legendre problem is as hard as the standard quadratic residuosity problem (Corollary 3.6). □

4.2 Analysis of the Legendre Pseudorandom Generator

We precisely define the secret-prime Legendre pseudorandom generator:

Definition 4.4 (Generalized Hidden-Prime Legendre PRG). Let $\lambda \in \mathbb{N}$ be a security parameter. For a sequence length ℓ and a vector $\vec{a} = (a_1, \ldots, a_\ell) \in \mathbb{Z}^\ell$, we define the length-$\ell$ generalized hidden-prime Legendre-sequence function $L_{\vec{a},\ell} \colon (\mathsf{BPrimes}_\lambda \times [2^\lambda]\}) \to \{-1,1\}^\ell$ as the function

$$L_{\vec{a},\ell}(p,x) \coloneqq \left[\left(\frac{x+a_1}{p}\right), \left(\frac{x+a_2}{p}\right), \ldots, \left(\frac{x+a_\ell}{p}\right) \right] \in \{-1,1\}^\ell.$$

Then, for any polynomial $\ell = \ell(\lambda)$, we define the set

$$\mathcal{L}_{\lambda,\ell} = \left\{ L_{\vec{a},\ell} \mid \vec{a} \in [2^{3\lambda}]^\ell \right\}.$$

We define $\mathcal{L}_\ell^{\mathsf{PRG}} = \{\mathcal{L}_{\lambda,\ell} \mid \lambda \in \mathbb{N}\}$ to be the generalized hidden-prime Legendre pseudorandom generator with output length ℓ.

Comparison to Damgård's PRG Damgård's formulation of the Legendre sequence pseudorandom generator [10] is a special case of Definition 4.4 where each set $\mathcal{L}_{\lambda,\ell}$ is a *singleton* set containing the function $L_{\vec{a},\ell}(p,x)$ where $\vec{a} = (1,2,\ldots,\ell)$ is a fixed vector, and where we also allow the input p to be any λ-bit prime. In this work, we consider the function family with random large offsets $\vec{a} \in [2^{3\lambda}]^\ell$ and the input prime p is restricted to a Blum prime (i.e., $p \equiv 3 \bmod 4$). We then reduce the problem of distinguishing the generalized hidden-prime Legendre pseudorandom generator to the standard quadratic-residuosity problem. Extending our results to the fixed-offset setting of Damgård's original paper remains an interesting open problem.

Our main theorem is the following:

Theorem 4.5 (Generalized Hidden-Prime Legendre PRG is Secure Under Quadratic Residuosity). *For all polynomials $\ell = \ell(\lambda)$ and for every efficient algorithm $\mathcal{A}$ that breaks pseudorandomness of $\mathcal{L}_\ell^{\mathsf{PRG}}$ with advantage* $\mathsf{PRGAdv}[\mathcal{A}](\lambda)$*, there exists an efficient algorithm $\mathcal{B}$ for the quadratic-residuosity problem with advantage* $\mathsf{QRAdv}_\ell[\mathcal{B}](\lambda)$ *and a negligible function* $\mathsf{negl}(\cdot)$ *where*

$$\mathsf{PRGAdv}[\mathcal{A}, \mathcal{L}_\ell^{\mathsf{PRG}}](\lambda) \leq \ell \cdot \mathsf{QRAdv}[\mathcal{B}](\lambda) + \mathsf{negl}(\lambda).$$

Theorem 4.5 follows by combining Theorem 4.2 with the following standard lemma:

Lemma 4.6 (Weak PRF Security Implies PRG Security). *For all polynomials $\ell = \ell(\lambda)$ and for every efficient algorithm $\mathcal{A}$ that breaks pseudorandomness of $\mathcal{L}_\ell^{\mathsf{PRG}}$ with advantage* $\mathsf{PRGAdv}[\mathcal{A}](\lambda)$*, there exists an efficient ℓ-query algorithm $\mathcal{B}$ that breaks weak pseudorandomness of $\mathcal{L}^{\mathsf{wPRF}}$ with advantage* $\mathsf{wPRFAdv}[\mathcal{B}](\lambda)$ *where*

$$\mathsf{wPRFAdv}[\mathcal{B}, \mathcal{L}^{\mathsf{wPRF}}] = \mathsf{PRGAdv}[\mathcal{A}, \mathcal{L}_\ell^{\mathsf{PRG}}](\lambda).$$

Proof. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ as follows:

- On input the function L_λ, Algorithm $\mathcal{B}$ makes ℓ queries to the weak PRF oracle to obtain values $(a_1, y_1), \ldots, (a_\ell, y_\ell)$.
- Algorithm $\mathcal{B}$ sets $\vec{a} = (a_1, \ldots, a_\ell)$ and $\vec{y} = (y_1, \ldots, y_\ell)$. It gives the function $L_{\vec{a},\ell}$ and the vector $\vec{y}$ to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs.

By definition, the weak PRF challenger samples $a_i \xleftarrow{R} [2^{3\lambda}]^\ell$ for all $i \in [\ell]$. This means the function $L_{\vec{a},\ell}$ is distributed uniformly over the set $\mathcal{L}_{\lambda,\ell}$. Consider now the distribution of $y_1, \ldots, y_\ell$:

- If $y_i = (\frac{x+a_i}{p})$ where $x \xleftarrow{R} [2^\lambda]$ and $p \xleftarrow{R} \mathsf{BPrimes}_\lambda$ are the components of the weak PRF key, then $\vec{y} = L_{\vec{a},\ell}(p, x)$. Thus, Algorithm $\mathcal{B}$ perfectly simulates the pseudorandom distribution for $\mathcal{A}$ and outputs 1 with probability $\rho_{\mathcal{A},0}(\lambda)$, where $\rho_{\mathcal{A},0}$ is the function from Definition 2.2.
- If $y_i \xleftarrow{R} \{-1, 1\}$, then Algorithm $\mathcal{B}$ perfectly simulates the truly random distribution for $\mathcal{A}$ and outputs 1 with probability $\rho_{\mathcal{A},1}(\lambda)$, where $\rho_{\mathcal{A},1}$ is the function from Definition 2.2.

We conclude that

$$\mathsf{wPRFAdv}[\mathcal{B}, \mathcal{L}^{\mathsf{wPRF}}] = |\rho_{\mathcal{A},0}(\lambda) - \rho_{\mathcal{A},1}(\lambda)| = \mathsf{PRGAdv}[\mathcal{A}, \mathcal{L}^{\mathsf{PRG}}]. \qquad \square$$

5 Open Questions

This paper leaves a number of questions open:

- Prior work assumes the security of the Legendre pseudorandom function when instantiated with a public prime modulus [2,3,13,16,18,22]. How does this assumption relate to other number-theoretic assumptions?
- Is there a better-than-exponential-time classical attack on the Legendre pseudorandom generator, in either the public- or secret-prime setting? There is a quantum polynomial-time attack on the Legendre pseudorandom function [16], given superposition queries [24] to the function. This attack does not apparently affect the Legendre pseudorandom generator.
- Given a quadratic-residuosity oracle, is there an efficient attack on the Legendre pseudorandom generator? Prior work [9] shows an efficient attack on the Jacobi pseudorandom function, given a factoring oracle.
- Is it possible to construct a key-exchange protocol from the assumption that Legendre sequences are pseudorandom?

Acknowledgments. We thank Brent Waters for insightful discussions about the quadratic-residuosity assumption and in particular, the proof of Theorem A.2. This work was funded in part by NSF and gifts from Capital One, Facebook, Google, Microsoft, Mozilla, NASDAQ, Seagate, and MIT's FinTech@CSAIL Initiative.

References

[1] N.S. Aladov. Sur la distribution des résidus quadratiques et non-quadratiques d'un nombre premier p dans la suite $1, 2, \ldots, p-1$. *Matematicheskii Sbornik*, 18(1):61–75, 1896.

[2] Marshall Ball, Justin Holmgren, Yuval Ishai, Tianren Liu, and Tal Malkin. On the complexity of decomposable randomized encodings, or: How friendly can a garbling-friendly PRF be? In *ITCS*, pages 86:1–86:22, 2020.

[3] Ward Beullens, Tim Beyne, Aleksei Udovenko, and Giuseppe Vitto. Cryptanalysis of the Legendre PRF and generalizations. *IACR Trans. Symmetric Cryptol.*, 2020(1):313–330, 2020.

[4] Ward Beullens and Cyprien Delpech de Saint Guilhem. LegRoast: Efficient post-quantum signatures from the Legendre PRF. In *PQCrypto*, pages 130–150, 2020.

[5] Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo-random number generator. *SIAM J. Comput.*, 15(2):364–383, 1986.

[6] Dan Boneh and Richard J. Lipton. Algorithms for black-box fields and their application to cryptography (extended abstract). In *CRYPTO*, pages 283–297, 1996.

[7] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In *CRYPTO*, pages 1–20, 2010.

[8] David A Burgess. The distribution of quadratic residues and non-residues. *Mathematika*, 4(2):106–112, 1957.

[9] Henry Corrigan-Gibbs and David J. Wu. The one-wayness of Jacobi signatures. In *CRYPTO*, 2024.

[10] Ivan Damgård. On the randomness of Legendre and Jacobi sequences. In *CRYPTO*, pages 163–172, 1988.

[11] Harold Davenport. On the distribution of quadratic residues (mod p). *Journal of the London Mathematical Society*, 1(1):49–54, 1931.

[12] Harold Davenport. On the distribution of quadratic residues (mod p). *Journal of the London Mathematical Society*, 1(1):46–52, 1933.

[13] Dankard Feist. Legendre pseudo-random function, 2019. `https://legendreprf.org/`.

[14] Paul Frixons and André Schrottenloher. Quantum security of the Legendre PRF. *Mathematical Cryptology*, 1(2):52–69, 2021.

[15] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*, pages 365–377, 1982.

[16] Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P. Smart. MPC-friendly symmetric key primitives. In *ACM CCS*, pages 430–443, 2016.

[17] Richard H Hudson. On the first occurrence of certain patterns of quadratic residues and non-residues. *Israel Journal of Mathematics*, 44(1):23–32, 1983.

[18] Novak Kalujerović, Thorsten Kleinjung, and Dušan Kostić. Cryptanalysis of the generalised Legendre pseudorandom function. In *Algorithmic Number Theory Symposium*, 2020.

[19] Dmitry Khovratovich. Key recovery attacks on the Legendre PRFs within the birthday bound. *IACR Cryptol. ePrint Arch.*, 2019.

[20] Rene Peralta. On the distribution of quadratic residues and nonresidues modulo a prime number. *Mathematics of Computation*, 58(197):433–440, 1992.

[21] René Peralta and Eiji Okamoto. Faster factoring of integers of a special form. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 79(4), 1996.

[22] István András Seres, Máté Horváth, and Péter Burcsi. The Legendre pseudorandom function as a multivariate quadratic cryptosystem: Security and applications. *IACR Cryptol. ePrint Arch.*, 2021.

[23] Victor Shoup. *A computational introduction to number theory and algebra.* Cambridge University Press, 2006.

[24] Wim van Dam and Sean Hallgren. Efficient quantum algorithms for shifted quadratic character problems. *arXiv preprint quant-ph/0011067*, 2000.

[25] Brent Waters, 2024. Personal communication.

[26] André Weil. On some exponential sums. *Proceedings of the National Academy of Sciences*, 34(5), 1948.

[27] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *FOCS*, pages 80–91, 1982.

A Hardness of Quadratic Residuosity for Blum Integers

Goldwasser and Micali [15] defined the quadratic-residuosity assumption relative to a standard RSA modulus $N = pq$. In this work, we consider the formulation by Blum, Blum, and Shub [5] where N is a Blum integer. A folklore claim (c.f., [7, Footnote 7]) says that because the Blum integers have constant density among random RSA modulus, the quadratic-residuosity assumption relative to RSA moduli implies the quadratic-residuosity assumption for Blum integers.

Since the implication was not immediate to us, we include a formal proof of this statement here for completeness. This proof was described to us by Brent Waters [25] and we are grateful that he has allowed us to include it here.

A.1 An Algorithm that Solves QR Modulo Blum Integers Might Not Solve QR Modulo Arbitrary RSA Moduli

We first show that an algorithm that solves quadratic residuosity with respect to a Blum integer might not not simultaneously solve quadratic residuosity with respect to a random RSA modulus. To illustrate this, we show that there could exist an efficient algorithm $\mathcal{A}$ that has advantage 1 at solving quadratic residuosity modulo Blum integers and advantage 0 at solving quadratic residuosity modulo arbitrary RSA moduli.

One such algorithm $\mathcal{A}$ could have the following behavior:

– Let $\nu(\lambda) < 1/2$ be the density of Blum integers among RSA moduli that are the product of λ-bit primes. Suppose that on input a Blum integer N and a challenge x,
  - if $x \in \mathbb{QR}_N$, algorithm $\mathcal{A}$ outputs 1, and
  - if $x \in \mathbb{J}_N \smallsetminus \mathbb{QR}_N$, algorithm $\mathcal{A}$ outputs 0.

  In particular, Algorithm $\mathcal{A}$ has advantage 1 for solving quadratic residuosity with respect to Blum integers.

– Take any positive constant $\varepsilon < 1 - 2\nu(\lambda)$. Suppose that on input a non-Blum-integer $N = pq$ and a challenge x,
  - if $x \in \mathbb{QR}_N$, Algorithm $\mathcal{A}$ outputs 1 with probability $\varepsilon/(1-\nu(\lambda))$, and
  - if $x \in \mathbb{J}_N \smallsetminus \mathbb{QR}_N$, Algorithm $\mathcal{A}$ outputs 1 with probability $(\nu(\lambda)+\varepsilon)/(1-\nu(\lambda))$.

If we then consider the behavior of Algorithm $\mathcal{A}$ on a random RSA modulus, which might or might not be a Blum integer, we find that:

– if $x \in \mathbb{QR}_N$, Algorithm $\mathcal{A}$ outputs 1 with probability

$$\rho_{\mathcal{A},0}(\lambda) = \nu(\lambda) \cdot 1 + (1 - \nu(\lambda)) \cdot \varepsilon/(1 - \nu(\lambda)) = \nu(\lambda) + \varepsilon,$$

  and

– if $x \in \mathbb{J}_N \smallsetminus \mathbb{QR}_N$, Algorithm $\mathcal{A}$ outputs 1 with probability

$$\rho_{\mathcal{A},1}(\lambda) = \nu(\lambda) \cdot 0 + (1 - \nu(\lambda)) \cdot (\nu(\lambda) + \varepsilon)/(1 - \nu(\lambda)) = \nu(\lambda) + \varepsilon = \rho_{\mathcal{A},0}(\lambda).$$

As such, the advantage of Algorithm $\mathcal{A}$ for breaking quadratic residuosity with respect to a random RSA modulus is 0, even though it has advantage 1 at breaking quadratic residuosity with respect to a Blum integer. In other words, because quadratic residuosity is a *decisional* assumption, the behavior of $\mathcal{A}$ on non-Blum-integers N can potentially exactly counteract the advantage of $\mathcal{A}$ on Blum integers, yielding an adversary with no overall advantage.

A.2 Waters' Proof that QR Modulo Blum Integers is as Hard as QR Modulo Arbitrary RSA Moduli

We now give a proof that quadratic residuosity modulo Blum integers is indeed as hard as quadratic residuosity modulo arbitrary RSA moduli. This proof was described to us by Brent Waters [25].

Definition A.1 (Quadratic Residuosity for RSA Moduli). Let Primes_λ be the set of λ-bit primes. For an algorithm $\mathcal{A}$, security parameter λ, and bit $b \in \{0,1\}$, we define $\rho_{\mathcal{A},b}(\lambda)$ to be

$$\rho_{\mathcal{A},b}(\lambda) := \Pr\left[\mathcal{A}(N, x_b) = 1 \;:\; \begin{array}{c} p, q \xleftarrow{\mathrm{R}} \mathsf{Primes}_\lambda \\ N \leftarrow pq \\ x_0 \xleftarrow{\mathrm{R}} \mathbb{QR}_N, x_1 \xleftarrow{\mathrm{R}} \mathbb{J}_N \smallsetminus \mathbb{QR}_N \end{array}\right].$$

We define the *quadratic-residuosity advantage with respect to RSA moduli* of an algorithm $\mathcal{A}$ as:

$$\mathsf{QRAdv}_{\mathsf{RSA}}[\mathcal{A}](\lambda) \coloneqq \left|\rho_{\mathcal{A},0}(\lambda) - \rho_{\mathcal{A},1}(\lambda)\right|.$$

Theorem A.2 (QR for RSA moduli implies QR for Blum integers). *Let $\delta(\lambda) \coloneqq \frac{|\mathsf{BPrimes}_\lambda|}{|\mathsf{Primes}_\lambda|}$ be the density of Blum primes. Then, for every efficient quadratic-residuosity adversary $\mathcal{A}$ with respect to Blum integers, there exists an efficient quadratic-residuosity adversary $\mathcal{B}$ with respect to RSA moduli where*

$$\mathsf{QRAdv}_{\mathsf{RSA}}[\mathcal{B}](\lambda) \geq \delta(\lambda)^2 \cdot \mathsf{QRAdv}[\mathcal{A}](\lambda).$$

Proof. Let $\mathcal{A}$ be an efficient quadratic-residuosity adversary with respect to Blum integers. Let $\mathsf{NonBPrimes}_\lambda = \mathsf{Primes}_\lambda \setminus \mathsf{BPrimes}_\lambda$ be the set of non-Blum-primes (i.e., the primes p where $p \equiv 1 \bmod 4$). For a bit $b \in \{0,1\}$, we define the following two quantities:

$$\rho^{\mathsf{Blum}}_{\mathcal{A},b}(\lambda) \coloneqq \Pr\left[\mathcal{A}(N, x_b) = 1 \ : \ \begin{array}{c} p, q \xleftarrow{\text{R}} \mathsf{BPrimes}_\lambda \\ N \leftarrow pq \\ x_0 \xleftarrow{\text{R}} \mathbb{QR}_N, x_1 \xleftarrow{\text{R}} \mathbb{J}_N \setminus \mathbb{QR}_N \end{array}\right]$$

$$\rho^{\mathsf{NonBlum}}_{\mathcal{A},b}(\lambda) \coloneqq \Pr\left[\mathcal{A}(N, x_b) = 1 \ : \ \begin{array}{c} p, q \xleftarrow{\text{R}} \mathsf{NonBPrimes}_\lambda \\ N \leftarrow pq \\ x_0 \xleftarrow{\text{R}} \mathbb{QR}_N, x_1 \xleftarrow{\text{R}} \mathbb{J}_N \setminus \mathbb{QR}_N \end{array}\right].$$

By definition,

$$\mathsf{QRAdv}[\mathcal{A}](\lambda) = \left|\rho^{\mathsf{Blum}}_{\mathcal{A},0}(\lambda) - \rho^{\mathsf{Blum}}_{\mathcal{A},1}(\lambda)\right|.$$

For $\beta \in \{0,1\}$, we define an Algorithm $\mathcal{B}_\beta$ as follows:

– Algorithm $\mathcal{B}_\beta$ receives a quadratic-residuosity challenge (N, x) from the challenger. First, if $N \equiv 3 \bmod 4$, then Algorithm $\mathcal{B}_\beta$ outputs 0.
– Otherwise, if $N \equiv 1 \bmod 4$, then Algorithm $\mathcal{B}_\beta$ runs Algorithm $\mathcal{A}$ on input $(N, (-1)^\beta \cdot x)$ and outputs whatever $\mathcal{A}$ outputs.

Certainly, if Algorithm $\mathcal{A}$ is efficient, then Algorithm $\mathcal{B}_\beta$ is efficient. We now compute the advantage of Algorithm $\mathcal{B}_\beta$.

– Suppose the challenger samples $x \xleftarrow{\text{R}} \mathbb{QR}_N$. We consider three subcases:
  - Suppose $N \equiv 3 \bmod 4$, then Algorithm $\mathcal{B}_\beta$ always outputs 0.
  - Suppose $N = pq$ and $p \equiv q \equiv 1 \bmod 4$. In this case, $-1 \in \mathbb{QR}_N$, so the distribution of x is $\mathsf{Uniform}(\mathbb{QR}_N)$. In this case, Algorithm $\mathcal{B}_\beta$ outputs 1 with probability $\rho^{\mathsf{NonBlum}}_{\mathcal{A},0}(\lambda)$.
  - Suppose $N = pq$ and $p \equiv q \equiv 3 \bmod 4$. In this case, $-1 \in \mathbb{J}_N \setminus \mathbb{QR}_N$. Thus, if $\beta = 0$, then the distribution of x is $\mathsf{Uniform}(\mathbb{QR}_N)$ whereas if $\beta = 1$, then the distribution of x is $\mathsf{Uniform}(\mathbb{J}_N \setminus \mathbb{QR}_N)$. We conclude that Algorithm $\mathcal{B}_\beta$ outputs 1 with probability $\rho^{\mathsf{Blum}}_{\mathcal{A},\beta}(\lambda)$.

  We conclude then that Algorithm $\mathcal{B}_\beta$ outputs 1 with probability

$$\rho_{\mathcal{B},0}(\lambda) = (1-\delta(\lambda))^2 \cdot \rho^{\mathsf{NonBlum}}_{\mathcal{A},0}(\lambda) + \delta(\lambda)^2 \cdot \rho^{\mathsf{Blum}}_{\mathcal{A},\beta}(\lambda).$$

– Suppose $x \in \mathbb{J}_N \setminus \mathbb{QR}_N$. We consider the same three subcases:
  - Suppose $N \equiv 3 \bmod 4$, then Algorithm $\mathcal{B}_\beta$ always outputs 0.
  - Suppose $N = pq$ and $p \equiv q \equiv 1 \bmod 4$. Then the distribution of x is $\mathsf{Uniform}(\mathbb{J}_N \setminus \mathbb{QR}_N)$, so Algorithm $\mathcal{B}_\beta$ outputs 1 with probability $\rho_{\mathcal{A},1}^{\mathsf{NonBlum}}(\lambda)$.
  - Suppose $N = pq$ and $p \equiv q \equiv 3 \bmod 4$. If $\beta = 0$, the distribution of x is $\mathsf{Uniform}(\mathbb{J}_N \setminus \mathbb{QR}_N)$ whereas if $\beta = 1$, then the distribution of x is $\mathsf{Uniform}(\mathbb{QR}_N)$. We conclude that Algorithm $\mathcal{B}_\beta$ outputs 1 with probability $\rho_{\mathcal{A},1-\beta}^{\mathsf{Blum}}(\lambda)$.

  We conclude then that Algorithm $\mathcal{B}_\beta$ outputs 1 with probability

$$\rho_{\mathcal{B},1}(\lambda) = (1-\delta(\lambda))^2 \cdot \rho_{\mathcal{A},1}^{\mathsf{NonBlum}}(\lambda) + \delta(\lambda)^2 \cdot \rho_{\mathcal{A},1-\beta}^{\mathsf{Blum}}(\lambda).$$

Next, define the functions

$$\begin{aligned} \varepsilon_{\mathsf{Blum}}(\lambda) &\coloneqq \rho_{\mathcal{A},0}^{\mathsf{Blum}}(\lambda) - \rho_{\mathcal{A},1}^{\mathsf{Blum}}(\lambda) \\ \varepsilon_{\mathsf{NonBlum}}(\lambda) &\coloneqq \rho_{\mathcal{A},0}^{\mathsf{NonBlum}}(\lambda) - \rho_{\mathcal{A},1}^{\mathsf{NonBlum}}(\lambda). \end{aligned}$$

Then, the advantage of Algorithm $\mathcal{B}_0$ and $\mathcal{B}_1$ can be written as

$$\begin{aligned} \mathsf{QRAdv}_{\mathsf{RSA}}[\mathcal{B}_0](\lambda) &= |\rho_{\mathcal{B}_0,0}(\lambda) - \rho_{\mathcal{B}_0,1}(\lambda)| \\ &= \left|(1-\delta(\lambda))^2 \cdot \varepsilon_{\mathsf{NonBlum}}(\lambda) + \delta(\lambda)^2 \cdot \varepsilon_{\mathsf{Blum}}(\lambda)\right| \\ \mathsf{QRAdv}_{\mathsf{RSA}}[\mathcal{B}_1](\lambda) &= |\rho_{\mathcal{B}_1,0}(\lambda) - \rho_{\mathcal{B}_1,1}(\lambda)| \\ &= \left|(1-\delta(\lambda))^2 \cdot \varepsilon_{\mathsf{NonBlum}}(\lambda) - \delta(\lambda)^2 \cdot \varepsilon_{\mathsf{Blum}}(\lambda)\right|. \end{aligned}$$

Since the sign of $\varepsilon_{\mathsf{NonBlum}}$ either matches the sign of $\varepsilon_{\mathsf{Blum}}$ or the sign of $-\varepsilon_{\mathsf{Blum}}$, we conclude that there exists $\beta \in \{0,1\}$ such that

$$\mathsf{QRAdv}_{\mathsf{RSA}}[\mathcal{B}_\beta] \geq \delta(\lambda)^2 \cdot |\varepsilon_{\mathsf{Blum}}(\lambda)| = \delta(\lambda)^2 \cdot \mathsf{QRAdv}[\mathcal{A}](\lambda). \qquad \square$$

Finally, as $p \to \infty$, the density of Blum primes tends to $\frac{1}{2}$ (c.f., [23, Theorem 5.21]). Combined with Theorem A.2, this shows that hardness of the quadratic residuosity assumption with respect to general RSA moduli implies hardness of quadratic residuosity with respect to Blum integers.