CRYPTANALYSIS OF THE GENERALISED LEGENDRE PSEUDORANDOM FUNCTION

NOVAK KALUĐEROVIĆ, THORSTEN KLEINJUNG, AND DUŠAN KOSTIĆ

Abstract. Linear Legendre pseudorandom functions were introduced in 1988 by Damgård, and higher degree generalisations were introduced by Russell and Shparlinski in 2004. We present new key recovery methods that improve the state of the art for both cases. For degree $r \geq 3$ we give an attack that runs in time $O(p^{r-3})$ after $O(p^3)$ precomputation for the most relevant high degree case; it is based on the action of the group of Möbius transformations on degree r polynomials. For $r < 3$ we give an $O(p^{r/2})$ attack with $O(p^{r/4})$ oracle queries. In the linear case we recovered the keys for the 64, 74 and 84-bit prime Ethereum challenges, being the first to solve the 84-bit case.

1. Introduction

The usage of Legendre symbols in a pseudorandom function (PRF) is an idea originally proposed by Damgård [3]. Further generalisations with higher degree polynomials were proposed by Russell and Shparlinski [9]. In both cases a prime p is given and the Legendre PRF is modelled as an oracle $\mathcal{O}$ that on input x outputs the Legendre symbol $\left(\frac{f(x)}{p}\right)$, where $f(x) \in \mathbb{F}_p[x]$ is a secret key. Damgård conjectured that when f is linear, given a sequence of Legendre symbols of consecutive elements it is hard to predict the next one. Similar problems conjectured to be hard were also proposed [7], such as finding the secret polynomial while being given access to $\mathcal{O}$ and distinguishing $\mathcal{O}$ from a random function. So far no polynomial time algorithms were found for either of these problems and it is believed that they are hard. Until recently practical applications have been limited, primarily due to availability of much faster alternatives.

A recent result by Grassi et al. [7] sparked an interest in the linear Legendre PRF because it was found suitable as a multi-party computation (MPC) friendly pseudorandom generator. This is mainly due to the homomorphic property of the Legendre symbol and the possibility of evaluating it with only three modular multiplications in arithmetic circuit multi-party computations, which makes it a very efficient MPC friendly PRF candidate.

There are plans to use this construction as a PRF for a proof of custody scheme in the Ethereum blockchain [6]. The proof of custody scheme requires a "*mix*" function i.e., a pseudorandom function that produces one bit of output. For this purpose the Legendre PRF was shown to be a great candidate because of its efficiency. In comparison, SHA256 requires tens of thousands of multiplications while AES needs 290 in the MPC setting [6].

Date: 21. February 2020.
2010 *Mathematics Subject Classification.* Primary 11T71.
Key words and phrases. Legendre PRF, cryptanalysis, group action.

In order to raise interest for this construction, Ethereum research posted a number of challenges [6]. The goal is to recover the secret key given 2^{20} consecutive Legendre symbols, for primes of size varying from 64 to 148 bits.

1.1. **Contribution.** In this paper we analyse the action of the group of Möbius transformations on monic polynomials of degree r, and we use it to give an improved attack on the Legendre pseudorandom function. For polynomials of degree $r \geq 3$ modulo a prime p we distinguish three types of polynomials and for the most relevant case we give an $O(p^{r-3})$ attack after an $O(p^3)$ precomputation with p oracle queries. For degree $r < 3$ an $O(p^{r/2})$ attack is given with $p^{r/4}$ queries. If the number of queries M is limited, we give an $O(\frac{p^r \log p}{M^2})$ attack. These are improvements with respect to the previous algorithms [2],[8] of factor from p up to p^3 in the general case, even higher for a new family of *bad* keys. In the linear and limited query case a factor of $\log p$ fewer trials in the search phase are needed.

We also give the solutions to challenges $0, 1$ and 2 of the Ethereum research linear Legendre PRF for 64, 74 and 84-bit primes. In all cases we were given access to 2^{20} Legendre symbols.

2. Background

Let p be an odd prime. Throughout the paper we suppose that the prime is public[1]. We denote with $\mathbb{F}_p$ the field of cardinality p.

2.1. Notation.

Definition 2.1 (Pseudorandom Functions)**.** A pseudorandom function family $\{\mathcal{O}_k\}_k$ is a set of functions with the same domain and codomain indexed by a set of keys k such that a function $\mathcal{O}_k$ chosen randomly over the set of k-values cannot be distinguished from a random function.

Definition 2.2 (Legendre symbol)**.** We define the Legendre symbol by setting

$$\left(\frac{x}{p}\right) = x^{\frac{p-1}{2}} = \begin{cases} 1 & \text{if } x \in \mathbb{F}_p \text{ is a square mod } p \\ -1 & \text{if } x \in \mathbb{F}_p \text{ is not a square mod } p. \end{cases}$$

In general the Legendre symbol is defined by setting $\left(\frac{0}{p}\right) = 0$, which makes the symbol multiplicative. However this comes at a cost of increasing the size of the codomain. In practice $\left(\frac{0}{p}\right) = 1$ is used.

We will assume that the multiplicative property of the Legendre symbol stands. This is a non-problem and the reader should be easily convinced that the algorithms we give terminate in the same expected time and with the same probability.

Definition 2.3 (Legendre sequence)**.** We define a Legendre sequence with starting point a and length L to be the sequence of Legendre symbols evaluated at L consecutive elements starting from a. We denote it with $\{a\}_L$:

$$\{a\}_L := \left(\frac{a}{p}\right), \left(\frac{a+1}{p}\right), \left(\frac{a+2}{p}\right), \ldots, \left(\frac{a+L-1}{p}\right).$$

[1]Originally, as proposed by Damgård, the prime was considered secret. We chose only to pursue the case of a public prime, as in the MPC use case.

Every a fully determines its sequence of length L, but not vice versa – that property depends on L. In general, these sequences are as well distributed as one can hope them to be. We know already that when $L = 1$ *"half"* of the a-values give 1, and the other *"half"* give -1. Similar properties are true for larger L, and in general, following a theorem of Davenport, around one in 2^L elements of $\mathbb{F}_p$ is a starting point of a given sequence of length L.

Theorem 2.4 (Davenport, 1933 [4])**.** *Let S be a finite sequence of ± 1's of length L. Then the number of elements of $\mathbb{F}_p$ whose sequence is equal to S satisfies*

$$\#\Big\{a \in \mathbb{F}_p \,\Big|\, \{a\}_L = S\Big\} = \frac{p}{2^L} + O(p^\varepsilon)$$

where $0 < \varepsilon < 1$ is a constant depending only on L.

Throughout the paper we assume that L is such that $\{a\}_L$ uniquely defines a, i.e., that the following holds

$$\{a\}_L = \{b\}_L \text{ if and only if } a = b. \tag{2.1}$$

It is easy to see that if we want this property to hold, we need $L = \Omega(\log_2 p)$. The only provable upper bound we have comes from the Weil bound [10] and is $L = O(\sqrt{p}\log p)$ which is exponential.

Our computational results, together with other statistical data on the distribution of Legendre sequences [3], indicate that on average over all sequences S of length L, there are $\frac{p}{2^L} + O(1)$ elements whose Legendre sequences are equal to S. In other words, for a random S and a random j we have $\{j\}_L = S$ with probability $\frac{1}{2^L}$. A good estimate of L in terms of p is $L = [2\log_2 p]$.

2.2. **The Legendre pseudorandom function.** In this section we define the Legendre pseudorandom function, and its higher degree generalisation.

Definition 2.5 (Legendre PRF)**.** The Legendre pseudorandom functions are functions $\mathcal{O}_k$ from $\mathbb{F}_p$ to $\{-1, 1\}$ indexed by $k \in \mathbb{F}_p$ and defined as

$$\mathcal{O}_k(x) = \left(\frac{x+k}{p}\right).$$

Definition 2.6 (Higher degree Legendre PRF)**.** The Legendre pseudorandom functions of degree r are a family of functions $\mathcal{O}_f$ from $\mathbb{F}_p$ to $\{-1, 1\}$ indexed by $f = k_r x^r + \ldots + k_1 x + k_0 \in \mathbb{F}_p[x]$ and defined as

$$\mathcal{O}_f(x) = \left(\frac{f(x)}{p}\right).$$

The degree r is assumed to be polylogarithmic in p.

Two oracles $\mathcal{O}_f(x)$ and $\mathcal{O}_{f/k_r}(x)$ are the same up to multiplication by $\left(\frac{k_r}{p}\right)$ and therefore we can assume the polynomial f to be monic. The case of linear $f(x)$ reduces to the standard Legendre PRF which we thus from now on refer to as the linear Legendre PRF.

The polynomial $f(x)$ is considered up to multiplication by a square since the Legendre symbol is invariant under square factors of $f(x)$. This is not entirely true as a square linear factor introduces a zero and may change the output of the oracle at one point, but the reader should be convinced that this can be safely ignored.

The secret key space, i.e., the space from which we choose $f(x)$ is the space of monic polynomials modulo squares. The number of such polynomials equals $p^r - p^{r-1}$ for $r > 1$ (see [1], problem 3.3) and p for $r = 1$.

Definition 2.7 (Generalised Legendre sequence)**.** The length L Legendre sequence of a polynomial $f(x)$ is denoted by $\{f\}_L$ and defined as

$$\{f\}_L := \left(\frac{f(0)}{p}\right), \left(\frac{f(1)}{p}\right), \left(\frac{f(2)}{p}\right), \ldots, \left(\frac{f(L-1)}{p}\right).$$

As a generalisation to Theorem 2.4 and property (2.1) we assume that L is such that $\{f\}_L$ uniquely defines f, i.e., that the following holds

$$\{f\}_L = \{g\}_L \text{ if and only if } f = g. \tag{2.2}$$

With r the degree of f we have $L = \Omega(r \log p)$. We assume that property (2.2) holds for $L = \Theta(r \log p)$. A reasonable estimate is $L = [2r \log p]$. Throughout the paper we include the dependence on L in the complexity of our algorithms.

2.3. **Hard Problems.** There are three main problems conjectured to be hard, and on which the security of the Legendre PRF is based.

Definition 2.8 (Generalised Legendre Symbol Problem - GLSP)**.** Let f be a uniformly random monic square-free polynomial. Given access to an oracle $\mathcal{O}$ that on input $x \in \mathbb{F}_p$ computes $\mathcal{O}(x) = \left(\frac{f(x)}{p}\right)$, find f.

Definition 2.9 (Decisional Generalised Legendre Symbol Problem - DGLSP)**.** Let f be a uniformly random monic square-free polynomial. Let $\mathcal{O}_0$ be an oracle that on input $x \in \mathbb{F}_p$ computes $\mathcal{O}_0(x) = \left(\frac{f(x)}{p}\right)$, and let $\mathcal{O}_1$ be an oracle that on input x outputs a random value in $\{-1, +1\}$. Given access to $\mathcal{O}_b$ where b is an unknown random bit, find b.

Definition 2.10 (Next Symbol Problem - NSP)**.** Given a Legendre sequence $\{f\}_M$ of $M = \text{polylog}(p)$ symbols, find $\left(\frac{f(M)}{p}\right)$, or equivalently find $\{f\}_{M+1}$.

It is easy to see that the GLSP and NSP are at least as hard as DGLSP. In the other direction, following a theorem of Yao [11] on general pseudorandom functions, predicting the next bit of a pseudorandom function is as hard as distinguishing it from a truly random one. Therefore $NSP = DGLSP \leq GLSP$, under polynomial time reductions.

3. Group action on polynomials

Möbius transformations act naturally on rational functions of $\mathbb{P}^1$, changing the argument and preserving their degrees. We show how this action can be exploited in order to connect oracles of monic polynomials that are in the same orbit.

3.1. **Möbius transformations.** Let $\mathcal{M}$ be the group of $\mathbb{F}_p$-rational automorphisms of $\mathbb{P}^1$. It is known that $\mathcal{M}$ is isomorphic to $\mathrm{PGL}_2(\mathbb{F}_p)$ and that this group has order $p^3 - p$. The elements of $\mathcal{M}$ are Möbius transformations. Given a matrix $m = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{PGL}_2(\mathbb{F}_p)$ there is a unique Möbius transformation φ_m given by

$$\begin{aligned} \varphi_m : \mathbb{P}^1 &\longrightarrow \mathbb{P}^1 \\ [x : y] &\longmapsto [ax + by : cx + dy], \end{aligned}$$

and function composition satisfies $\varphi_{m_1} \circ \varphi_{m_2} = \varphi_{m_1 m_2}$. We drop the notion of φ_m and only use m from now on.

3.2. **Action of $\mathcal{M}$ on monic polynomials.** The action of a Möbius transformation $m = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathcal{M}$ on a polynomial f is denoted by $m \cdot f = f_m$ and defined as

$$m \cdot f = f_m(x) \coloneqq f\left(\frac{ax+b}{cx+d}\right)\frac{(cx+d)^r}{f(\frac{a}{c})c^r}. \tag{3.1}$$

The corrective factors $(cx+d)^r$ and $f(\frac{a}{c})c^r$ are introduced in order to make f_m a polynomial and to make it monic correspondingly.

There is another way to look at this action – if α is a root of f then $m^{-1}(\alpha)$ is a root of f_m, where $m^{-1} = \left(\begin{smallmatrix} d & -b \\ -c & a \end{smallmatrix}\right)$ is the inverse of the Möbius transformation m. Thus, if $f(x) = \prod_{i=1}^{r}(x-\alpha_i)$ then

$$f_m(x) = \prod_{i=1}^{r}(x - m^{-1}\alpha_i) = \prod_{i=1}^{r}(x - \frac{d\alpha_i - b}{-c\alpha_i + a}). \tag{3.2}$$

Therefore the group $\mathcal{M}$ of Möbius transformations has left (covariant) action on the roots of polynomials in $\mathbb{F}_p[x]$ and right (contravariant) action on polynomials.

3.3. **Obtaining oracles of polynomials in the orbit.** Suppose we are given access to $\mathcal{O}$, the oracle of f. Following (3.1) we can mimic the oracle of f_m with

$$\left(\frac{f_m(x)}{p}\right) = \mathcal{O}\left(\frac{ax+b}{cx+d}\right)\left(\frac{cx+d}{p}\right)^r \mathcal{O}\left(\frac{a}{c}\right)\left(\frac{c}{p}\right)^r.$$

Therefore we can obtain $\{f_m\}_L$ by computing $L+1$ Legendre symbols and querying the oracle $L+1$ times. If $c = 0$ then $\mathcal{O}(\frac{a}{c})(\frac{c}{p})^r$ is substituted with $(\frac{a}{p})^r$. If $cx+d = 0$ for some $x \in [0, L)$, then we substitute $\mathcal{O}(\frac{ax+b}{cx+d})(\frac{cx+d}{p})^r$ by $(\frac{ax+b}{p})^r$.

3.4. **Polynomial types.** We divide the key space into three sets based on reducibility of the polynomials and the size of their orbit given by the action of $\mathcal{M}$. The following lemma helps characterise these sets.

Lemma 3.1. *Let $\mathcal{M} = \mathrm{PGL}_2(\mathbb{F}_p)$ and $f \in \mathbb{F}_p[x]$ an irreducible polynomial of degree r with $3 \le r < p$. Then, the stabiliser of f is a cyclic group of order r' for some $r' \mid r$. Furthermore $r' \mid p^2 - 1$.*

Proof. Let $\mathrm{Stab}(f) = \{m \in \mathcal{M} | f = f_m\}$ be the stabiliser of f, and let $m \in \mathrm{Stab}(f)$. By property (3.2) the roots of f_m are $m^{-1}\alpha_i$ implying that m permutes the roots of f. Let $\mathrm{Gal}(f) = \{\phi_i \coloneqq x \mapsto x^{p^i} | i \in \mathbb{Z}/r\}$ be the Galois group of f, and let α be any root of f. Then $m\alpha = \phi_i(\alpha)$ for some $i \in \mathbb{Z}/r$. Furthermore $m(\phi_j(\alpha)) = \phi_j(m\alpha) = \phi_j(\phi_i(\alpha)) = \phi_i(\phi_j(\alpha))$ since m is rational and it commutes with the Frobenius. Therefore each element of the stabiliser acts on the roots as an element of $\mathrm{Gal}(f)$. This gives rise to a homomorphism from $\mathrm{Stab}(f)$ to $\mathrm{Gal}(f)$ which is injective since two Möbius transformations with the same action on a set of $r \ge 3$ points have to be equal. Therefore $\mathrm{Stab}(f)$ is a subgroup of $\mathrm{Gal}(f) \cong \mathbb{Z}/r$, so it is isomorphic to $\mathbb{Z}/r'$ for some $r' \mid r$. The stabiliser is naturally a subgroup of $\mathcal{M}$, so its order divides $\#\mathcal{M} = p(p^2-1)$. Since $r' < p$ we have $r' \mid p^2 - 1$. □

Definition 3.2. We call irreducible polynomials with a trivial stabiliser "*good*", irreducible polynomials with a stabiliser of size $r' > 1$ are called "*bad*", and reducible polynomials are called "*ugly*".

4. Algorithm

We give an algorithm for solving the Generalised Legendre Symbol Problem. We start by querying the oracle $\mathcal{O}(x)$ at all $x \in \mathbb{F}_p$, and computing $\left(\frac{x}{p}\right)$ for all $x \in \mathbb{F}_p$. These results are then saved in a table and whenever we need an oracle query or a Legendre symbol we read them instead of computing an expensive symbol or querying the oracle multiple times.

The general idea is to do a table-based collision search. We make a table containing $\{f_m\}_L$ for some $m \in \mathcal{M}$, and we try random g until $\{g\}_L = \{f_m\}_L$ for some m. This gives us $f = g_{m^{-1}}$. The tables and the trials differ for different polynomial types, so we give three separate algorithms for *good*, *bad* and *ugly* polynomials.

4.1. Good polynomials algorithm.

We recall that f is *good* if it is an irreducible polynomial of degree $r \geq 3$ and the stabiliser of f is trivial.

4.1.1. *Precomputation.* In the precomputation stage we generate a table T containing $\{f_m\}_L$ and a description of m for all Möbius transformations m as described in Section 3.3. Since f is *good*, the table T contains $p^3 - p$ different sequences.

4.1.2. *Search.* The search is done by trying random $g(x)$ of degree r and computing $\{g\}_L$ until we find a hit, which we expect to find after $O(p^{r-3})$ trials. For each trial g is evaluated at L points, and L Legendre symbols are extracted, so the run time can be measured in the number of Legendre symbols extracted which is $O(p^{r-3}L)$.

4.2. Bad polynomials algorithm.

We recall that f is *bad* if it is an irreducible polynomial of degree $r \geq 3$ and the stabiliser of f is non-trivial. It follows from Lemma 3.1 that $\mathrm{Stab}(f)$ is isomorphic to $\mathbb{Z}/r'$.

4.2.1. *Precomputation.* We start by finding $\mathrm{Stab}(f)$, the stabiliser of f. A straightforward way to find it in $O(p^3)$ is by enumerating $\mathcal{M}$ and isolating the matrices that fix f. Appendix A describes a non-trivial way to find it in $O(p^2 \log r)$ steps.

Call m any generator of $\mathrm{Stab}(f)$. The matrix m is rational so it has a Jordan canonical form of one of the following three types:

$$\begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \qquad \begin{pmatrix} \lambda & 0 \\ 0 & \mu \end{pmatrix} \qquad \begin{pmatrix} a & 1 \\ 0 & a \end{pmatrix}$$

Type 1 Type 2 Type 3

where $a, b \in \mathbb{F}_p \setminus \{0\}$ and $\lambda, \mu \in \mathbb{F}_{p^2} \setminus \mathbb{F}_p$, conjugates of each other. We can exclude Type 3 matrices since they have order p, while m has order $r' < p$.

Let D be a diagonal matrix of order r' and P a change of basis matrix (these can be chosen uniquely from a set of representatives given in Appendix A) such that

$$m = P\, D\, P^{-1}.$$

Following from $D \cdot f_P = (PD) \cdot f = (mP) \cdot f = P \cdot f_m = P \cdot f = f_P$, the polynomial f_P is stabilised by D. Therefore f_P satisfies $f_P(\frac{r}{s}x)(\frac{s}{r})^r = f_P(\frac{r}{s}x) = f_P(x)$ where $(r, s) = (a, b)$ or (λ, μ). This sets the following constraints on the coefficients of f_P

$$f_P(x) = x^r + k_{r-1}x^{r-1} + \ldots + k_2x^2 + k_1x + k_0 = x^r + \sum_{i=0}^{r-1} k_i x^i$$

$$(D \cdot f_P)(x) = x^r + k_{r-1}(\tfrac{r}{s})^{r-1}x^{r-1} + \ldots + k_1(\tfrac{r}{s})x + k_0 = x^r + \sum_{i=0}^{r-1} k_i(\tfrac{r}{s})^i x^i$$

from which it follows that

$$k_i = k_i(\tfrac{r}{s})^i \text{ for } i = 0, 1, \ldots, r-1. \tag{4.1}$$

Since $\frac{r}{s}$ has order r' we have $k_i = 0$ for all i that are not multiples of r'.

We create a table T of size $O(p)$ containing polynomials t in the orbit of f with t_P satisfying (4.1). The process differs for the two types of matrices so we treat them separately.

Type 1. When D is rational, P is rational too, so the polynomial f_P is in the orbit of f. If C is a rational diagonal matrix, $C \cdot f_P$ is another polynomial in the orbit of f satisfying (4.1). The total number of such polynomials is $\frac{p-1}{r'}$ since matrices C can be chosen up to stabiliser of f_P which is $\langle D \rangle$. A set of representatives is

$$\mathcal{C}_1 = \left\{ \begin{pmatrix} g^i & 0 \\ 0 & 1 \end{pmatrix} \middle| g \in \mathbb{F}_p^* \text{ generator}, 0 \le i < \tfrac{p-1}{r'} \right\}. \tag{4.2}$$

The table T contains $\{PCP^{-1} \cdot f\}_L$ together with a description of C for all C in $\mathcal{C}_1$. It has $\frac{p-1}{r'}$ elements, and for all polynomials t in the table, t_P satisfies (4.1).

Type 2. When D is irrational, P is too, so f_P is not in the orbit of f. There are additional constraints on f_P following from the rationality of m:

$$m = P \begin{pmatrix} \lambda & 0 \\ 0 & \mu \end{pmatrix} P^{-1} = \overline{m} = \overline{P} \, \overline{\begin{pmatrix} \lambda & 0 \\ 0 & \mu \end{pmatrix}} \, \overline{P}^{-1} = \overline{P} \begin{pmatrix} \mu & 0 \\ 0 & \lambda \end{pmatrix} \overline{P}^{-1}.$$

Let $A_P := P^{-1}\overline{P}$. From the definition of A_P and the above formulas it follows that

$$A_P^{-1} = \overline{A}_P$$
$$\begin{pmatrix} \lambda & 0 \\ 0 & \mu \end{pmatrix} A_P = A_P \begin{pmatrix} \mu & 0 \\ 0 & \lambda \end{pmatrix}.$$

These constraints imply that $A_P = \left(\begin{smallmatrix} 0 & \alpha \\ 1/\overline{\alpha} & 0 \end{smallmatrix}\right)$ for some $\alpha \in \mathbb{F}_{p^2}$. The action of A_P is the same as the action of $\left(\begin{smallmatrix} 0 & s \\ 1 & 0 \end{smallmatrix}\right)$ where $s = \alpha\overline{\alpha} \in \mathbb{F}_p$. Note that s can be computed and, up to choosing a different representative for P, can be set to be equal to 1. We further have

$$A_P \cdot f_P(x) = f_{PA_P}(x) = f_{\overline{P}}(x) = \overline{P} \cdot f(x) = \overline{P} \cdot \overline{f(x)} = \overline{P \cdot f(x)} = \overline{f_P(x)},$$

which gives new constraints on the coefficients of $f_P(x)$:

$$\overline{f_P(x)} = x^r + \overline{k}_{r-1}x^{r-1} + \ldots + \overline{k}_2 x^2 + \overline{k}_1 x + \overline{k}_0 = x^r + \sum_{i=0}^{r-1} \overline{k}_i x^i$$

$$(A_P \cdot f_P)(x) = x^r + \tfrac{k_1 s}{k_0}x^{r-1} + \ldots + \tfrac{k_{r-1}s^{r-1}}{k_0}x + \tfrac{s^r}{k_0} = x^r + \sum_{i=0}^{r-1} \tfrac{k_{r-i}s^{r-i}}{k_0}x^i.$$

This translates to

$$\begin{aligned} k_0^{p+1} &= s^r \\ k_{r-i} &= \frac{k_0 \overline{k}_i}{s^{r-i}} \\ k_{r/2}^{p-1} &= \frac{s^{r/2}}{k_0} \text{ if } r \text{ is even.} \end{aligned} \tag{4.3}$$

The polynomial f_P is not the only polynomial satisfying (4.1) and (4.3). Certainly (4.1) is satisfied for every $C \cdot f_P$ where C is a diagonal matrix. In order for $C \cdot f_P$ to satisfy (4.3) we need $A_P \cdot f_{PC} = \overline{f_{PC}}$, which implies

$$(CA\overline{C}^{-1}) \cdot f_P(x) = \overline{f_P(x)}.$$

This condition, together with C being diagonal implies that C is contained in

$$\left\{ \begin{pmatrix} c & 0 \\ 0 & \overline{c} \end{pmatrix} \middle| c \in \mathbb{F}_{p^2}^* \right\}.$$

Multiplying C on the right by a rational scalar matrix or by an element of $\mathrm{Stab}(f_P) = \langle D \rangle$ does not change the polynomial $C \cdot f_P$. Therefore C can be chosen from a reduced set of representatives, for example the following:

$$\mathcal{C}_2 = \left\{ \begin{pmatrix} g^i & 0 \\ 0 & \overline{g}^i \end{pmatrix} \middle| g \text{ generator of } \mathbb{F}_{p^2}^*,\ 0 \le i < \tfrac{p+1}{r''} \right\},$$

where $\frac{p+1}{r''} = \gcd(p+1, \frac{p^2-1}{r'})$, in other words $r'' = \frac{r'}{\gcd(r', p-1)}$. The choice of r'' follows from the exponents of g being chosen modulo $p+1$ (action of $\mathbb{F}_p^*$) and modulo $\frac{p^2-1}{r'}$ (action of r''th roots of unity).

The table T contains $\{PCP^{-1} \cdot f\}_L$ together with a description of C for all C in $\mathcal{C}_2$ (note that PCP^{-1} is rational). It has $\frac{p+1}{r''}$ elements, and for all polynomials t in the table, t_P satisfies (4.1) and (4.3).

4.2.2. *Search.* In the search phase we go over $g(x) = x^r + \sum_{i=0}^{r/r'-1} g_i x^i$ that satisfy (4.1) and compute $\{g_{P^{-1}}\}_L$ until we find a hit in T. In that case $f = g_{(PC)^{-1}}$.

For Type 1, the coefficients g_i are in $\mathbb{F}_p$. The total number of polynomials g is $p^{r/r'}$ and we expect to find a hit after $O(p^{r/r'-1}r')$ trials.

For Type 2, the coefficients g_i are in $\mathbb{F}_{p^2}$ and they satisfy (4.3). Therefore there are $p+1$ choices for g_0, the g_i with $1 \le i < r/2$ can be chosen freely, giving p^2 choices each, and the g_j for $r/2 < j$ are constrained to one value for each choice of the previous coefficients. If r is even, $g_{r/2}$ has $p-1$ choices. The total number of polynomials g is $O(p^{r/r'})$ and we expect to find a hit after $O(p^{r/r'-1}r'')$ trials.

4.3. **Ugly polynomials algorithm.** We recall that f is *ugly* if it is a reducible polynomial of degree $r \ge 3$. Write $f(x) = l(x)h(x)$ where $r_h = \deg(h(x)) \ge r/2$.

The Legendre symbol is multiplicative, and Möbius transformations are homomorphic with respect to polynomial multiplication, so we have $\{f_m\}_L = \{l_m\}_L\{h_m\}_L$, where the multiplication is element-wise. It follows that $\{f_m\}_L\{l_m\}_L = \{h_m\}_L$.

4.3.1. *Precomputation.* We create two tables, T_1 containing $\{f_m\}_L$ for all $m \in \mathcal{M}$, and T_2 containing sequences of all polynomials $g(x)$ of degree $r - r_h$ (the candidates for $l_m(x)$). The main table T is a product of T_1 and T_2, i.e., a table of size $O(p^{r-r_h+3})$ containing $\{f_m\}_L\{g\}_L$ for all $m \in \mathcal{M}$ and all g.

4.3.2. *Search.* The search phase constitutes of trying random polynomials $t(x)$ of degree r_h until we find a hit in T. This gives $\{t\}_L = \{f_m\}_L\{g\}_L$, and implies that $t(x) = h_m(x)$, $g(x) = l_m(x)$, and finally $f(x) = g_{m^{-1}}(x)t_{m^{-1}}(x)$. We expect to find a solution in $O(p^{r_h-3})$ trials.

The above description glosses over a number of minor details that one needs to be careful about. The run time is actually p^{r_h} divided by the size of the orbit of $h(x)$.

If h is *good*, then its orbit is maximal and we are good.

If h is *bad*, we can test all *bad* h in time $O(p^{r_h/r'}L)$ for each $r'_h \mid r_h$, so in total $O(p^{r_h/2}L)$. For both Type 1 and Type 2 we can enumerate all polynomials h in time $O(p^{r_h/r'_h-1}r''_hL)$ with r''_h defined as in (4.2).

If h is *ugly*, we analyse two cases:

1.) h has an irreducible factor of degree at least 3.
Suppose $h = h_1h_2$ of degrees r_1 and r_2. We select a set of $O(p^{r_1-3})$ representatives for h_1, multiply them with polynomials of degree r_2 and search for $\{h\}_L = \{h_1\}_L\{h_2\}_L$ in T, achieving an $O(p^{r_h-3})$ run time.

2.) h has all factors of degree ≤ 2.
There are three subcases to consider:
- h is divisible by a product of three linear polynomials. Then at least one h_m is divisible by $x(x-1)(x-2)$, so we test for $h = x(x-1)(x-2)h_2$ where h_2 are of degree $r_h - 3$.
- h is divisible by a linear and quadratic polynomial. Then one of h_m is divisible by $x(x^2-u)$ where u is a chosen non-square, so we test for $h = x(x^2-u)h_2$ where h_2 are of degree $r_h - 3$.
- h is divisible by two quadratic polynomials. Then one of them can be considered to be x^2-u where u is a non-square, and the other one has only 1 degree of freedom. We test for $h = (x^2-u)h_1h_2$ where h_1 is selected from $O(p)$ quadratic polynomials and h_2 is of degree $r_h - 4$.

Therefore if f is *ugly* we can find it in $O(p^{r_h-3})$ trials irrespective of the type of h.

TABLE 1. Comparisons of best known algorithms for solving the degree $r \geq 3$ Legendre PRF, in big-O's. Size of the stabiliser of f is denoted with r', and $r'' = r'$ if $r' \mid p-1$ and $r'' = r'/\gcd(r', p-1)$ otherwise. We denote with r_h the degree of a factor of f which is at least $r/2$. Complexity is given in the number of Legendre symbols computed/extracted. In all cases we need p queries.

good polynomials	search	precomputation	memory
Khovratovich [8]	$p^{r-1}r\log p$	$r\log p$	$r\log p$
Beullens et al. [2]	$p^{r-2}r^2\log^2 p$	p^2	p^2
Our algorithm	$p^{r-3}r\log p$	$p^3r\log p$	$p^3r\log p$
bad polynomials	search	precomputation	memory
Khovratovich [8]	$p^{r-1}r\log p$	$r\log p$	$r\log p$
Beullens et al. [2]	$p^{r-2}r^2\log^2 p$	p^2	$p^{r-r_h}r\log p$
Our algorithm	$p^{r/r'-1}r''r\log p$	$p^2r\log p$	$(p/r'')r\log p$
ugly polynomials	search	precomputation	memory
Khovratovich [8]	$p^{r-1}r\log p$	$r\log p$	$r\log p$
Beullens et al. [2]	$p^{r_h}r\log p$	$p^{r-r_h}r\log p$	$p^{r-r_h}r\log p$
Our algorithm	$p^{r_h-3}r\log p$	$p^{r-r_h+3}r\log p$	$p^{r-r_h+3}r\log p$

4.4. **Time-memory tradeoff for low degrees.** The run time of the algorithm depends mainly on the search stage. However for some low degree polynomials,

the precomputation may take longer than the search stage. In some cases a time-memory tradeoff allows to reduce the complexity further.

4.4.1. *Good polynomials.* For $r \geq 6$, the table-based collision search with an $O(p^3)$ table and $O(p^{r-3})$ trials is optimal. For $3 \leq r \leq 5$, a tradeoff with an $O(p^{r/2})$ table and $O(p^{r/2})$ trials is better.

4.4.2. *Bad polynomials.* If $r/r' - 1 < 2$ then the bottleneck is the precomputation phase that takes $O(p^2 \log r)$ steps. This can happen when $r' = r/c$ for $c = 1, 2$. Not much can be done to reduce the precomputation cost since testing *badness* costs $O(p^2 \log r)$. For $r = 3$ we can lower the attack complexity to $O(p^{1.5})$ with table-based collision search for *good* polynomials.

4.4.3. *Ugly polynomials.* We test if f is ugly by trying to find it using the *ugly* polynomials algorithm for each $r_h = \lceil r/2 \rceil, \ldots, r-1$. The precomputation cost is $O(p^{r-r_h+3})$ and the search cost is $O(p^{r_h-3})$.

If $r - r_h + 3 > r_h - 3$, i.e., $r_h < r/2 + 3$, then we can do a tradeoff. Call $\varepsilon := r_h - r/2 < 3$. We compute only the action of p^ε many matrices on f, and after multiplying with the table T_2 of p^{r-r_h} sequences, obtain a table of size $p^{r/2}$. We expect to finish the search phase in $O(p^{r_h-\varepsilon}) = O(p^{r/2})$ if a collision exists. Otherwise we assume that f does not have a factor of degree r_h and move to $r_h + 1$.

4.5. **Security recommendations.** Following our argumentation, the most secure PRFs are the ones coming from *good* polynomials. While we can test for irreducibility in polynomial time, the only way to distinguish *good* and *bad* polynomials is by means of the $O(p^2 \log r)$ algorithm from Appendix A. The number of *bad* polynomials is small, and can be shown to be bounded from above by

$$\sum_{\substack{r' \mid \gcd(r, p^2-1) \\ r' > 1}} -\mu(r') p^{r/r'+1} r' = O(p^{r/2+1} r).$$

The easiest way to assure that our secret polynomial is not *bad* is to choose p and r such that $\gcd(r, p^2 - 1) = 1$.

4.6. **Degree $r = 2$.** If $r = 2$ all polynomials are *bad* or *ugly*. There is a deterministic $O(p)$ algorithm for finding f in this case – we first precompute the action of $\{\left(\begin{smallmatrix} 1 & a \\ 0 & 1 \end{smallmatrix}\right) | a \in \mathbb{F}_p\}$ on the polynomial f, which assures that the precomputed table contains the Legendre sequence of a polynomial of the form $x^2 - c$:

$$\left(\begin{smallmatrix} 1 & a \\ 0 & 1 \end{smallmatrix}\right) \cdot (x^2 - tx + n) = x^2 - (t - 2a)x + (n + a^2 - ta).$$

Then we test all p such polynomials until we find f.

5. Limited query case and the linear Legendre PRF

In the Section 4 we query the oracle at all elements of $\mathbb{F}_p$ and then extract up to $p^3 - p$ sequences. The reader should be convinced that the same argumentation works with $p - o(p/L)$ queries, as we still have access to $\Omega(p^3)$ sequences. When the secret polynomial is linear doing more than $O(p^{1/2}L)$ queries is wasteful. Indeed creating a table with $O(p^{1/2})$ sequences by doing L queries per sequence allows us to find the secret polynomial after $O(p^{1/2})$ trials. This is essentially the algorithm in [8], where the author further provides a memoryless approach.

The main difference in the linear case with respect to the higher degree case is that we are allowed $M \leq \sqrt{p}L$ queries to the oracle. How many different group actions can we obtain from only M queries? The same question can be asked in the higher degree case, and the algorithm we provide can be directly applied in that scenario. One would expect a cubic increase, as with full access to the oracle, but this seems to be out of reach.

5.1. **Linear shifts subgroup.** Let G be the subgroup of $\mathcal{M}$ consisting only of linear Möbius transformations,

$$G = \left\{ \begin{pmatrix} d & i \\ 0 & 1 \end{pmatrix} \middle| d \in \mathbb{F}_p^*, i \in \mathbb{F}_p \right\} \leqslant \mathrm{PGL}_2(\mathbb{F}_p).$$

An element $(i, d) := \left(\begin{smallmatrix} d & i \\ 0 & 1 \end{smallmatrix}\right)$ sends $f(x)$ to $f_{i,d}(x)$. In order to extract $\{f_{i,d}(x)\}_L$ from the oracle $\mathcal{O}$ of f, we compute

$$\left(\frac{f_{i,d}(x)}{p}\right) = \mathcal{O}\left(\frac{dx+i}{0x+1}\right)\left(\frac{0x+1}{p}\right)^r\left(\frac{d}{p}\right)^r = \mathcal{O}\left(dx+i\right)\left(\frac{d}{p}\right)^r$$

for all $x \in [0, L)$. If $\mathcal{O}$ is queried in $[0, M)$, then we can extract all $f_{i,d}$ such that $dx + i \in [0, M)$ for all $x \in [0, L)$. This creates the following constraints on i, d:

$$\begin{cases} d = 1, 2, \ldots, \left\lfloor \frac{M-1}{L-1} \right\rfloor \\ i = 0, 1, \ldots, M - 1 - (L-1)d \end{cases} \quad \text{or} \quad \begin{cases} d = -1, -2, \ldots, -\left\lfloor \frac{M-1}{L-1} \right\rfloor \\ i = (L-1)(-d), \ldots, M-1. \end{cases}$$

The total number of eligible $(i, d) \in G$ is

$$\sum_{d=1}^{\left\lfloor \frac{M-1}{L-1} \right\rfloor} 2(M - (L-1)d) = \frac{M^2}{L-1} - M + O(L)$$

with the constant in $O(L)$ being at most 2.

The limited query algorithm works as follows:

5.1.1. *Precomputation.* Query $\mathcal{O}$ at $[0, M)$. Extract $O(\frac{M^2}{L})$ Legendre sequences $\{f_{i,d}\}_L$ and save them in a table T together with descriptions of (i, d).

5.1.2. *Search.* Search is done by trying random polynomials until we find a hit in the table, which is expected after $O(\frac{p^r L}{M^2})$ trials, in particular $O(\frac{pL}{M^2})$ for the linear PRF.

5.1.3. *Further improvements.* The cost of the precomputation is M queries and $O(\frac{M^2}{L})$ *sequence extractions*. The cost of the search is $O(\frac{pL}{M^2})$ *trials*. A straightforward way to do a *sequence extraction* is to read the pre-saved queries L times. Due to the nature of the sequences, this cost can be amortised to $O(1)$ per sequence. Doing a *trial* constitutes of evaluating the polynomial in L places and computing L Legendre symbols. Again, this cost can be amortised to $O(\log L)$ per trial. These implementational improvements are not within the scope of this paper, and they are explained in detail in [5].

5.2. **Algorithm comparison.** The first algorithm by Khovratovich [8] computes sequences with on-the-go queries, and directly computes Legendre symbols. The main benefit of this approach is that it is memoryless. This was improved on in [2] by extracting sequences rather than querying/computing symbols, and increasing the sequence yield to M^2/L^2. In our terminology, the authors of [2] use the same group G but only elements (i, d) such that $i < d$, leading them to a table which is a factor of L smaller with respect to ours. Using the full group G as in 5.1 comes with cheaper sequence extraction in the precomputation stage, but more expensive sequence extraction in the search stage thus the $\log \log p$ factor in Table 2. A more detailed analysis is given in [5].

TABLE 2. Comparisons of best known algorithms for the linear Legendre PRF challenge, in big-O's and $\Theta(\log p)$-bit word operations. We denote with t the time to compute a Legendre symbol

Algorithm	search	precomputation	memory	optimal run time
Khovratovich [8]	$\frac{p\, t \log^2 p}{M}$	M	$\log p$	$\sqrt{p}\, t \log p$
Beullens et al. [2]	$\frac{p \log^2 p}{M^2}$	M^2	$\frac{M^2}{\log p}$	$\sqrt{p} \log p$
Our algorithm	$\frac{p \log p \log \log p}{M^2}$	$\frac{M^2}{\log p}$	M^2	$\sqrt{p \log \log p}$

5.3. **Experiments.** Ethereum research posted a number of challenges [6] for breaking the linear Legendre PRF. In each challenge we are given a prime p of size varying from 64 to 148 bits, and $M = 2^{20}$ bits of the sequence $\{k\}_M$ as defined in Section 2.3. The challenge is to recover the key k. In each case we were able to precompute a table with $\sim 2^{34}$ sequences. The most interesting is of course challenge #2 since it had not been solved before. The actual number of trials performed in challenge #2 is $2^{46.97} = 1.38e14$ which is far less than expected. This can be explained by large variance and by sheer luck. The two most difficult challenges (#3 and #4) are out of reach with the proposed attack and its implementation. An in-depth explanation of the experiments is given in [5]. The code and the keys of the first three challenges can be found at

`https://github.com/nKolja/LegendrePRF`.

TABLE 3. Results and estimates for solving the Legendre PRF challenges [6].

Challenge	Prime bit size	Expected # trials	Observed # trials	Expected core-hours	Observed core-hours
0	64	2^{30}	$2^{30.78}$	290 sec	490 sec
1	74	2^{40}	$2^{39.53}$	82	59
2	84	2^{50}	$2^{46.97}$	1.4e5	1.72e4
3	100	2^{66}	-	9.1e9	-
4	148	2^{114}	-	2.5e24	-

References

1. Elwyn R. Berlekamp, *Algebraic coding theory - revised edition*, World Scientific Publishing Co., Inc., USA, 2015.
2. Ward Beullens, Tim Beyne, Aleksei Udovenko, and Giuseppe Vitto, *Cryptanalysis of the Legendre PRF and generalizations*, Cryptology ePrint Archive, Report 2019/1357, 2019, `https://eprint.iacr.org/2019/1357`.
3. Ivan Damgård, *On the randomness of Legendre and Jacobi sequences*, Proceedings of the 8th Annual International Cryptology Conference on Advances in Cryptology (London, UK), CRYPTO '88, Springer-Verlag, 1990, pp. 163–172.
4. H. Davenport, *On the distribution of quadratic residues (mod p)*, Journal of the London Mathematical Society **s1-8** (1933), no. 1, 46–52.
5. Novak Kaluđerović, Thorsten Kleinjung, and Dušan Kostić, *Improved key recovery on the Legendre PRF*, Cryptology ePrint Archive, Report 2020/098, 2020, `https://eprint.iacr.org/2020/098`.
6. Dankard Feist, *Legendre pseudo-random function*, 2019, `https://legendreprf.org/bounties`.
7. Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P. Smart, *MPC-friendly symmetric key primitives*, Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (New York, NY, USA), CCS '16, ACM, 2016, pp. 430–443.
8. Dmitry Khovratovich, *Key recovery attacks on the Legendre PRFs within the birthday bound*, Cryptology ePrint Archive, Report 2019/862, 2019, `https://eprint.iacr.org/2019/862`.
9. Alexander Russell and Igor E. Shparlinski, *Classical and quantum function reconstruction via character evaluation*, Journal of Complexity **20** (2004), no. 2-3, 404–422 (English).
10. André Weil, *On some exponential sums*, Proceedings of the National Academy of Sciences **34** (1948), no. 5, 204–207.
11. Andrew C. Yao, *Theory and application of trapdoor functions*, Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (USA), SFCS 82, IEEE Computer Society, 1982, p. 8091.

Appendix A. Computing the stabiliser $\mathrm{Stab}(f)$ of f

Let $m \in \mathrm{Stab}(f)$ be a matrix of order r'. Following the same argumentation from Section (4.2) there exists a change of coordinate matrix P such that $D = P^{-1}mP$ is a diagonal matrix. We give a set of representatives for matrices D and P such that for each m there is a single pair D, P in that set satisfying

$$m = P\, D\, P^{-1}.$$

This property can be used to argue that we need only to find one m_r of order p_r for any prime divisor $p_r \mid r'$. Given m_r, an element m_i of order p_r^i is simply $P\sqrt[i]{D}P^{-1}$, and an element m_q of order q_r for some other divisor $q_r \mid r'$ is PD_qP^{-1} for the corresponding matrix D_q of order q_r. Furthermore, an element of order p_rq_r can be found by computing $m_r^u m_q^v$ with $up_r + vq_r = 1$. Therefore in order to find the full stabiliser group we need only to find one element of prime order. This is done by searching for elements of order q in the stabiliser, for each prime $q \mid r$, so we assume that we know r'.

The search for m is done by going through the conjugacy class of a matrix D of order r', until we find a matrix that stabilises f. The conjugacy class has size $\Theta(p^2)$ so we expect to find m in p^2 steps, but we have to be careful and go through the whole class without repetitions.

The process is explained separately for Type 1 and Type 2 matrices.

A.1. Matrices of Type 1.

If m is of Type 1 then for some $P \in \mathrm{GL}_2(\mathbb{F}_p)$

$$m = P \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} P^{-1}$$

where $a, b \in \mathbb{F}_p$ non-zero such that $\xi := a/b$ has order r'. Since m is defined up to scalar multiplication in $\mathbb{F}_p^*$, we may suppose that $a = \xi$ and $b = 1$, so $D = \left(\begin{smallmatrix} \xi & 0 \\ 0 & 1 \end{smallmatrix}\right)$ for some ξ primitive r''th root of unity in $\mathbb{F}_p$. There are in total $\varphi(r')$ different ξ values to consider, however each one will give rise to a different generator of the stabiliser of f, so the choice of ξ does not matter.

The search for m is done by enumerating PDP^{-1}, where matrices P are chosen from $\mathrm{GL}_2(\mathbb{F}_p)$ up to right multiplication by an element of $Z(D) = \{\left(\begin{smallmatrix} a & 0 \\ 0 & b \end{smallmatrix}\right) | \, ab \neq 0\}$, the centraliser of D. In total there are $p^2 + p$ elements in $\mathrm{GL}_2(\mathbb{F}_p)/Z(D)$. One set of representatives can be chosen to be

$$\{\left(\begin{smallmatrix} 0 & 1 \\ 1 & d \end{smallmatrix}\right), \left(\begin{smallmatrix} 1 & 0 \\ c & 1 \end{smallmatrix}\right), \left(\begin{smallmatrix} 1 & 1 \\ c & d \end{smallmatrix}\right) | \;\; c, d \in \mathbb{F}_p \text{ such that the determinants are non-zero}\}.$$

When $r' = 2$, so $D = \left(\begin{smallmatrix} -1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$, the set of representatives is halved because $\left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right) \in Z(D)$ after projecting on $\mathrm{PGL}_2(\mathbb{F}_p)$. In that case we give the following of $(p^2+p)/2$ representatives for the matrices P :

$$\{\left(\begin{smallmatrix} 0 & 1 \\ 1 & d \end{smallmatrix}\right), \left(\begin{smallmatrix} 1 & 1 \\ c & d \end{smallmatrix}\right) | \;\; c < d \in \mathbb{F}_p\}$$

where the ordering of elements of $\mathbb{F}_p$ is induced from the lift to $\{0, 1, \ldots, p-1\}$.

A.2. **Matrices of Type 2.** If m is of Type 1 then for some $P \in \mathrm{GL}_2(\mathbb{F}_{p^2})$

$$m = P \begin{pmatrix} \lambda & 0 \\ 0 & \mu \end{pmatrix} P^{-1}$$

where $\lambda, \mu \in \mathbb{F}_{p^2}$ are conjugate roots of an irreducible second degree polynomial such that $\xi := \lambda/\mu$ is a primitive r''th root of unity.

Lemma A.1. *The diagonal matrix D defined above is unique in* $\mathrm{GL}_2(\mathbb{F}_{p^2})/\mathbb{F}_p^*$.

Proof. Since $\xi = \overline{\mu}/\mu = \mu^{p-1}$ we have $\xi^{p+1} = 1$. Due to the primitivity of ξ it follows that $r' \mid p + 1$.

If $\xi \in \mathbb{F}_p$ then $\xi^2 = 1$ so $\xi = -1$ and $r' = 2$. In that case $\lambda = -\mu$, so the minimal polynomial of λ is $x^2 - c$ for some non-square c. Up to multiplying D by a constant in $\mathbb{F}_p^*$, we may suppose $\lambda = \sqrt{u}$ for a fixed non-square u, and therefore there is only one such matrix.

If ξ is not rational, then $\overline{\xi} = \xi^p = 1/\xi$, so $\xi\overline{\xi} = 1$. From $\lambda = \xi\mu$ we have $D = \left(\begin{smallmatrix} \xi\mu & 0 \\ 0 & \mu \end{smallmatrix}\right)$. The determinant and the trace of D are the same as those of m, so in particular they are rational. This means that

$$\mu(\xi + 1) \in \mathbb{F}_p$$
$$\xi\mu^2 \in \mathbb{F}_p$$

from which it follows that $\mu = \frac{a}{\xi+1}$ and $\lambda = \frac{\xi a}{\xi+1}$ for some $a \in \mathbb{F}_p$. For any choice of a, the second condition follows from $\xi\overline{\xi} = 1$. Multiplying λ and μ by any non-zero rational constant does not change the property of D being conjugate to $m \in \mathrm{PGL}_2(\mathbb{F}_p)$, to them being irrational conjugates of each other or to their quotient being equal to ξ. Therefore we may suppose $\lambda = \frac{\xi}{\xi+1}$ and $\mu = \frac{1}{\xi+1}$. □

We start by computing a primitive root of unity ξ of order r', and set D as above. As before, the choice of ξ does not matter.

The search for m follows by going through $P\,D\,P^{-1}$ where the matrices P are chosen such that PDP^{-1} is rational and up to right multiplication by $Z(D)$, the centraliser of D.

A.2.1. *Rational PDP^{-1}.* If PDP^{-1} is rational we have $PDP^{-1} = \overline{P}\,\overline{D}\,\overline{P}^{-1}$, so

$$(P^{-1}\overline{P})\left(\begin{smallmatrix}\mu & 0\\ 0 & \lambda\end{smallmatrix}\right) = \left(\begin{smallmatrix}\lambda & 0\\ 0 & \mu\end{smallmatrix}\right)(P^{-1}\overline{P}).$$

Define $A_P := P^{-1}\overline{P}$. The matrix A_P satisfies $A_P^{-1} = \overline{A_P}$, so it has to satisfy

$$A_P = \begin{pmatrix} 0 & \alpha \\ 1/\overline{\alpha} & 0 \end{pmatrix}$$

for some non-zero α in $\mathbb{F}_{p^2}$. From $\overline{P} = PA_P$ we have some constraints on P,

$$P \in \left\{ \begin{pmatrix} q & \overline{q\alpha} \\ r & \overline{r\alpha} \end{pmatrix} \,\middle|\, q, r \in \mathbb{F}_{p^2},\ qr \neq 0,\ q^{p-1} \neq r^{p-1} \right\}.$$

A.2.2. *The centraliser $Z(D)$.* The matrix D is diagonal with different eigenvalues, so

$$Z(D) = \left\{ \begin{pmatrix} x & 0 \\ 0 & \overline{y} \end{pmatrix} \,\middle|\, x, y \in \mathbb{F}_{p^2},\ xy \neq 0 \right\}.$$

Multiplying a P on the right by an element of the centraliser gives

$$\begin{pmatrix} q & \overline{q}\,\overline{\alpha} \\ r & \overline{r}\,\overline{\alpha} \end{pmatrix} \begin{pmatrix} x & 0 \\ 0 & \overline{y} \end{pmatrix} = \begin{pmatrix} qx & \overline{q}\,\overline{\alpha}\,\overline{y} \\ rx & \overline{r}\,\overline{\alpha}\,\overline{y} \end{pmatrix} = \begin{pmatrix} qx & \overline{qx}\,\overline{\left(\frac{\alpha y}{x}\right)} \\ rx & \overline{rx}\,\overline{\left(\frac{\alpha y}{x}\right)} \end{pmatrix},$$

which sends (q, r) to (qx, rx) and α to $\alpha\frac{y}{x}$, so we may assume that $q = \alpha = 1$. A set of $p^2 - p$ representatives for matrices P is

$$\left\{ \begin{pmatrix} 1 & 1 \\ r & \overline{r} \end{pmatrix} \,\middle|\, r \in \mathbb{F}_{p^2} \setminus \mathbb{F}_p \right\}.$$

When $r' = 2$, so $D = \left(\begin{smallmatrix}\sqrt{u} & 0\\ 0 & -\sqrt{u}\end{smallmatrix}\right)$ for some rational non-square u, the set of representatives is halved because $\left(\begin{smallmatrix}0 & 1\\ 1 & 0\end{smallmatrix}\right) \in Z(D)$ after projecting on $\mathrm{GL}_2(\mathbb{F}_p^2)/\mathbb{F}_p^*$. In that case we give the following $(p^2 - p)/2$ representatives for matrices P:

$$\left\{ \begin{pmatrix} 1 & 1 \\ r & \overline{r} \end{pmatrix} \,\middle|\, r = a\sqrt{u} + b\,, \quad 1 \leq a \leq \tfrac{p-1}{2},\ 0 \leq b < p \right\}.$$

Laboratory for cryptologic algorithms, École polytechnique fédérale de Lausanne, Lausanne, Switzerland

Email address: novak.kaluderovic@epfl.ch

Email address: thorsten.kleinjung@epfl.ch

Email address: dusan.kostic@epfl.ch