Key recovery attacks on the Legendre PRFs within the birthday bound

Dmitry Khovratovich
khovratovich@gmail.com
Dusk and Evernym and ABDK Consulting and Sikoba
Supported by Ethereum Foundation

July 24, 2019

Abstract

We show that Legendre PRF, recently suggested as an MPC-friendly primitive in a prime field $\mathbb{Z}_p$, admits key recovery attacks of complexity $O(\sqrt{p})$ rather than previously assumed $O(p)$. We also demonstrate new attacks on high-degree versions of this PRF, improving on the previous results by Russell and Shparlinski.

1 Introduction

Pseudo-random function (PRF) is an important cryptographic primitive. Typically denoted $F_K(\cdot)$ with K being a secret key, its security is usually defined as inability to distinguish the output from a randomly chosen function f on the same domain by an adversary who does not know neither K nor f. Different PRF candidates have been proposed, with block ciphers like AES being the most secure examples. AES and other blockcipher-based PRF candidates with n-bit keys and inputs are assumed to be secure to distinguishing and key recovery attacks with complexity up to 2^n. In contrast, PRF candidates whose security is based on discrete logarithm hardness and similar assumptions typically claim security only up to the birthday bound and even less [DY05]. In this paper we show that the Legendre PRF candidate falls into the second category as it fails to provide security comparable to AES.

Legendre PRF The Legendre PRF has been introduced recently [Gra+16] as a MPC-friendly candidate as its multi-party computation requires only a few multiplications which are the bottleneck in many MPC implementations.

Let p be a prime and a a positive integer, then the Legendre symbol $L_p(a)$ is defined as

$$L_p(a) = a^{(p-1)/2} \bmod p$$

and denoted $\left(\dfrac{a}{p}\right)$. If $a = b^2$ for some b then $L_p(a) = 1$, otherwise $L_p(a) = -1$.

Damgard [Dam88], based on tests that demonstrate statistical uniformity of quadratic residues modulo p, suggested a keyed Legendre symbol $L_p^K(a) = L_p(K+a)$ as a pseudo-random generator outputting 1 or -1 by incrementing a. Damgard conjectured that no polynomially bounded adversary can recover K with reasonable probability given access to the oracle that computes $L_p^K(a)$ for any a, which became known as *Legendre hidden shift problem* [Gra+16]. A naive deterministic algorithm guesses K and compares the entire keystream of length p with the guessed one, thus spending p^2 time. Russell and Shparlinski [RS04] demonstrated, based on the Weil bound, that a deterministic algorithm may consider keystream segments as short as $\log^2 p$, thus bringing down the complexity to $p\log^2 p$. Note that a naive randomized algorithm, selecting a random a to start with, hopes to check the guess using only $\log p$ outputs and has total complexity of $p\log p$. Together with the Russell-Shparlinski bound, these are the best results on the Legendre keyed generator so far. One can also consider a high-degree generator

$$L_p^{K_0,K_1,\ldots,K_{d-1}}(a) = L_p(K_0 + K_1 a + K_2 a^2 + \cdots + K_{d-1}a^{d-1} + a^d)$$

with d keys. The Russell-Shparlinski deterministic algorithm requires $d^2 p^d \log^2 p$ operations, whereas a naive randomized algorithm needs $p^d \log p$ operations.

Our contributions We demonstrate new algorithms for key recovery in Legendre PRF, both in degree-1 and high-degree versions. Our attacks are based on time-memory tradeoff attacks and memoryless collision search algorithms.

2 Memoryless attack on the Legendre keyed generator

Here we consider the Legendre linear PRF

$$L^K(a) = L_p(K+a).$$

Let us denote for vector $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ the set of PRF evaluations

$$L^K(\mathbf{a}) = \left(L^K(a_1), L^K(a_2), \ldots, L^K(a_n)\right).$$

We first formalize the uniformity assumption that we use to filter out key candidates. Concretely, we assume that *for any vector* $\mathbf{a} = (a_1, a_2, \ldots, a_{\log p})$ *and any* $\log p$*-bit string* $\mathbf{b}$ *the number of keys* K *such that* $L^K(\mathbf{a}) = \mathbf{b}$ *is* $O(1)$. It is a very natural cryptanalytic assumption and it is also confirmed by statistical tests. A conservative attacker may use the Weil bound [RS04] which provably upper bounds the length of such strings by $\log^2 p$.

We then note that the Legendre PRF has a very simple related-key property that holds with probability 1. Indeed, for any $\delta \in \mathbb{Z}_p$:

$$L^K(a) = L^{K+\delta}(a-\delta).$$

Then we proceed as follows. Let N be an integer and $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ be a vector of $\mathbb{Z}_p$ elements.

1. Make N guesses of $K : K^1, K^2, \ldots, K^N$ and compute N vectors
$$V[K^i] = L^{K^i}(\mathbf{a}) = \left(L^{K^i}(a_1), L^{K^i}(a_2), \ldots, L^{K^i}(a_n)\right).$$
2. Select randomly N elements of $\mathbb{Z}_p$: $A^1, A^2, \ldots, A^N$ and make $N \cdot n$ queries to the PRF so that N vectors are stored:
$$W[A^i] = L^K(A^i + \mathbf{a}) = \left(L^K(A^i + a_1), L^K(A^i + a_2), \ldots, L^K(A^i + a_n)\right).$$
3. Suppose that $K^i - A^j = K$ for some i, j. Then
$$W[A^j] = L^K(A^j + \mathbf{a}) = L^{K^i - A^j}(A^j + \mathbf{a}) = L^{K^i}(\mathbf{a}) = V[K^i].$$
Therefore it suffices to find an intersection between $\{W[A^j]\}_j$ and $\{V[K^i]\}_i$.

If we denote $f(x) = V[x]$ and $g(y) = W[y]$, then the key recovery is equivalent to the collision search f and g. Thus $N = O(\sqrt{p})$ suffices.

A collision search between two functions can be done memoryless by first reducing the search to a single function h [MOM91] and then making a memoryless collision search. The single function is defined as:

$$h(x) = \begin{cases} f(x), & \text{if } \phi(x) = 1; \\ g(x), & \text{if } \phi(x) = 0. \end{cases}$$

where ϕ is some simple predicate like a XOR of all bits.

The overall complexity of the attack is $O(\sqrt{p}\log p)$ PRF queries and Legendre evaluations. If only $M < \sqrt{p}$ queries are available, then the attack costs $O(p(\log p)/M)$ computations. In the unlikely case we get too many false alarms, we can simply select another $\mathbf{a}$.

3 Quadratic Legendre PRF

Now we consider the polynomial version of Legendre PRF and start with degree 2:

$$L^{K_0,K_1}(a) = \left(\frac{K_0 + K_1 a + a^2}{p}\right).$$

A naive randomized algorithm just guesses K_0, K_1, computes $\log p$ outputs and compares with PRF queries. It has complexity $O(p^2 \log p)$. We can do better by guessing only K_1 and applying our attack on the linear case, with a simple replace of a with a^2. This algorithm has complexity $O(p^{1.5} \log p)$.

We can do better by recalling some attacks on stream ciphers. Babbage [Bab95] considered a clocked stream cipher with internal state of $\log N$ bits and showed that if we can make M queries to the cipher so that it changes state M times, then we should run the cipher starting at N/M random states and search a collision between guessed keystreams and the actual keystream.

Unfortunately, this attack does not apply directly since in our quadratic generator we do not have a state that evolves. If we set (K_0, K_1, a) then only a would change but neither K_0 nor K_1, so Babbage's attack does not seem to work.

To make the approach work we introduce another related-key property. Recall now that

$$L(a) = a^{(p-1)/2} \implies L(ab) = L(a)L(b).$$

Now let r be some integer, then $L(r^2) = 1$. We obtain

$$L^{K_0,K_1}(a) = L^{K_0,K_1}(a)L(r^2) = \left(\frac{K_0 + K_1 a + a^2}{p}\right)\left(\frac{r^2}{p}\right) = \\ = \left(\frac{K_0 r^2 + K_1 a r^2 + a^2 r^2}{p}\right) = L^{K_0 r^2, K_1 r}(ar).$$

Or equivalently

$$L^{K_0 r^2, K_1 r}(a) = L^{K_0,K_1}(a/r). \tag{1}$$

Thus we can compute the PRF on p related keys using p different r on the same input. However, we need $\log p$ inputs for each related key. We could use arbitrary $\log p$ values, but there is a better choice which allows reusing Legendre computation for another related key.

Concretely, consider $\mathbf{a} = (r, r^2, \ldots, r^n)$. Then

$$L^{K_0,K_1}(\mathbf{a}) = (L^{K_0,K_1}(r), L^{K_0,K_1}(r^2), \ldots, L^{K_0,K_1}(r^n))$$

and

$$L^{K_0 r^2, K_1 r}(\mathbf{a}) = (L^{K_0 r^2, K_1 r}(r), L^{K_0 r^2, K_1 r}(r^2), \ldots, L^{K_0 r^2, K_1 r}(r^n)) = \\ = (L^{K_0,K_1}(1), L^{K_0,K_1}(r), \ldots, L^{K_0,K_1}(r^{n-1}))$$

Therefore, querying the PRF on r^i for many i we obtain $L^{K_0 r^{2ri}, K_1 r^i}(\mathbf{a})$.

The full attack works as follows:

1. For N guesses of $K : K^1 = (K_0^1, K_1^1), K^2 = (K_0^2, K_1^2), \ldots, K^N = (K_0^N, K_1^N)$ and a vector $\mathbf{a} = (r, r^2, \ldots, r^n)$ compute

$$V[K^i] = L^{K^i}(\mathbf{a}) = \{L^{K^i}(r), L^{K^i}(r^2), \ldots, L^{K^i}(r^n)\}.$$

2. For N values $r, r^2, \ldots, r^N$ compute:

$$W[r^j] = L^{(K_0,K_1)\circ(r^{2j},r^j)}(\mathbf{a}) = \\ = \left(L^{(K_0,K_1)\circ(r^{2j},r^j)}(r), L^{(K_0,K_1)\circ(r^{2j},r^j)}(r^2), \ldots, L^{(K_0,K_1)\circ(r^{2j},r^j)}(r^n)\right).$$

3. If for some i, j we have $K^i = (K_0, K_1) \circ (r^{2j}, r^j)$ then $V[K^i] = W[r^j]$. We need p elements in each set to have a collision.

The attack can be done memoryless using the same approach as in Section 2. The overall complexity is $O(p \log p)$.

Generator	Rus-Shpa	Randomized	Ours
Linear $L^K()$	$p\log^2 p$	$p\log p$	$\sqrt{\mathbf{p}}\log\mathbf{p}$
Quadratic $L^{K_0,K_1}()$	$p^2\log^2 p$	$p^2\log p$	$\mathbf{p}\log\mathbf{p}$
High-deg $L^{K_0,K_1,\ldots,K_{d-1}}()$	$p^d d^2\log^2 p$	$p^d d\log p$	$\mathbf{p^{d-1}d}\log\mathbf{p}$

Table 1: Summary of our and previous results on the Legendre PRF

4 High-degree Legendre PRF

We finally consider a high-degree version:

$$L^{K_0,K_1,\ldots,K_{d-1}}(a) = \left(\frac{K_0 + K_1 a + \ldots + K_{d-1}a^{d-1} + a^d}{p}\right)$$

The attack is a simple reduction to the quadratic case:

1. Guess $K_2, K_3, \ldots, K_{d-1}$;
2. Apply Section 3 attack to K_0, K_1 with the modified property
$$L^{K_0 r^d, K_1 r^{d-1}}(a) = L^{K_0,K_1}(a/r^d)L(r^d). \tag{2}$$
3. The attack complexity is $O(p^{d-1}d\log p)$.

5 Future work

References

[Bab95] SH Babbage. "Improved exhaustive search attacks on stream ciphers". In: (1995) (cit. on p. 3).

[Dam88] Ivan Damgård. "On the Randomness of Legendre and Jacobi Sequences". In: *CRYPTO*. Vol. 403. Lecture Notes in Computer Science. Springer, 1988, pp. 163–172 (cit. on p. 2).

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. "A Verifiable Random Function with Short Proofs and Keys". In: *Public Key Cryptography*. Vol. 3386. Lecture Notes in Computer Science. Springer, 2005, pp. 416–431 (cit. on p. 1).

[Gra+16] Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, et al. "MPC-Friendly Symmetric Key Primitives". In: *IACR Cryptology ePrint Archive* 2016 (2016), p. 542 (cit. on pp. 1, 2).

[MOM91] Hikaru Morita, Kazuo Ohta, and Shoji Miyaguchi. "A Switching Closure Test to Analyze Cryptosystems". In: *CRYPTO*. Vol. 576. Lecture Notes in Computer Science. Springer, 1991, pp. 183–193 (cit. on p. 3).

[RS04] Alexander Russell and Igor E. Shparlinski. "Classical and quantum function reconstruction via character evaluation". In: *J. Complexity* 20.2-3 (2004), pp. 404–422 (cit. on p. 2).