OPRFs from Isogenies: Designs and Analysis

Lena Heimberger[1], Fredrik Meisingseth[1,2], Christian Rechberger[1]

[1]Graz University of Technology, [2] Know-Center
lena.heimberger@iaik.tugraz.at

May 5, 2023

Abstract

Oblivious Pseudorandom Functions are an elementary building block in cryptographic and privacy-preserving applications. However, while there are numerous pre-quantum secure OPRF constructions, few options exist in a post-quantum secure setting. Isogeny group actions and the associated low bandwidth seem like a promising candidate to construct a quantum-resistant OPRF. While there have been relevant attacks on isogeny-related hardness assumptions, the commutative CSIDH is unaffected. In this work, we propose OPUS, a novel OPRF with small communication complexity, requiring only CSIDH as the security assumption. Our results also revisit the Naor-Reingold OPRF from CSIDH and show how to efficiently compute offline evaluations. Additionally, we analyze a previous proposal of a CSIDH-based instantiation of the Naor- Reingold construction. We report several issues with the straightforward instantiation of the protocol and propose mitigations to address those shortcomings. Our mitigations require additional hardness assumptions and more expensive computations but result in a competitive protocol with low communication complexity and few rounds. Our comparison against the state of the art shows that OPUS and the repaired, generic construction are competitive with other proposals in terms of speed and communication size. More concretely, OPUS achieves almost two orders of magnitude less communication overhead compared to the next-best lattice-based OPRF at the cost of higher latency and higher computational cost.

1 Introduction

Cloud computing, authenticated key exchange and secure data sharing are ubiquitous in modern-day computation. All of these high-level applications may use Oblivious Pseudorandom Functions (OPRFs) as an underlying building block to heighten security and guarantee privacy. Informally, OPRFs take input from a client and a key from a server, then return a pseudorandom output to the client. Consider password-authenticated key agreement: Authenticating against

a server with a password usually requires transmitting the password, ideally in a salted, hashed form. However, passwords notoriously lack entropy and may be recovered in the event of a breach. In addition, attacks leaking cleartext passwords are still common, for example, the PwnedPasswords web page [Hun] lists at the time of writing this paper 91 matches when searching for *plain text* breaches. All of those breaches could be prevented by never storing passwords on a server. A great example a protocol solving this problem is OPAQUE, an asymmetric password-authenticated key agreement protocol for which standardization efforts are ongoing at the CFRG [DFHSW22]. However, OPRFs are not limited to passwords: Privacy Pass [DGS+18] uses OPRFs to reduce the number of CATCHPAs required to be solved by users to browse the internet. A related problem is private contact discovery [KRS+19], which protects the highly sensitive social graph of messenger app users from ever being uploaded to a server.

While there is a variety of sound and efficient constructions for classical primitives, OPRFs from post-quantum hardness assumptions remain an open question. To be more specific, almost all OPRFs are based on some form of the discrete logarithm problem or, more rarely, factoring [CHL22]. A nice primitve for quantum-resistant OPRFs are isogenies, which have small communication complexity, but suffer from slow runtimes. Until now, there was only one OPRF based on CSIDH [BKW20]. We show that the naïve approach to the implementation is not sufficient, and subsequently propose a fix using CSI-FiSh.

Based on the work on this OPRF, we report OPUS, a novel construction that efficiently computes the Naor-Reingold OPRF while only using 60% of the group actions of the previous proposal. On a practical side, we implement and benchmark our novel OPRF, concretizing the computational, communication, and round complexity of isogeny protocols. In addition, we revisit the Naor-Reingold PRF from CSIDH and discuss the security in-depth. Finally, we compare against the state of the art and find that OPUS is competitive with current proposals, as it requires no preprocessing, no trusted setup and has an implementation available.

2 Preliminaries

We start by defining (Oblivious) Pseudorandom Functions.

Definition 1 (Pseudorandom Function) *A PseudoRandom Function (PRF) [GGM84, GGM86] is a deterministic function F which takes two inputs K and X from distinct sets and maps them in polynomial time to the output N such that $F_K : \{0,1\}^k \times \{0,1\}^x \to \{0,1\}^n$. F is pseudorandom when there is no probabilistic polynomial-time algorithm to distinguish N from a randomly chosen output. F must be computable in polynomial time given the set members K and X.*

Definition 2 (Oblivious Pseudorandom Function) *An Oblivious Pseudorandom Function(OPRF) is a protocol between two parties. One party holds the*

secret key K and the other holds their secret input X. The OPRF privately realizes the joint computation outputting $F_K(X)$ to the party holding X, and nothing to the party holding K. [FIPR05]

2.1 CSIDH

CSIDH [CLM+18], was originally proposed as a quantum-safe replacement for Diffie-Hellman key exchanges. It builds on the ideas of Couveignes [Cou06] and Rostovtsev-Stolbunov [RS06](CRS), but restricts the isogeny graph to supersingular curves over $\mathbb{F}_p$. p is a prime in the form $p = 4\prod_{i=1}^{n} \ell_i - 1$ and $p \equiv 3 \mod 4$. For $\pi = \sqrt{-p}$ and $\mathcal{O} = \mathbb{Z}[\pi]$, each ℓ_i splits the endomorphism ring $\mathcal{O}$ into $\mathfrak{l}_i$ isogenies with degree ℓ_i. The isogeny $\phi : E \to E'$ is a map from an elliptic curve E to another curve E' that preserves the point at infinity and the algebraic structure [Sil86]. Hence, both curves have the same number of rational points. The isogeny is unique up to isomorphism. It is computed using Velu's formula [Vél71].

The heart of CSIDH is the group action $*$ acting on the set of elliptic curves $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$, denoted as $\mathcal{E}$. To ensure the group action is efficient, each ℓ_i is required to be a small, distinct, odd prime.

2.1.1 Checking if a curve is supersingular.

For a prime $p \geq 5$, an elliptic curve over $\mathbb{F}_p$ is supersingular if and only if the order of E is $p + 1$. Therefore, verification of supersingular curves is done by checking the order of the curve. This check has a small chance of erroneously classifying a supersingular curve as ordinary. More complex algorithms prevent the wrong classification at the cost of- higher memory and time complexity [Sut12].

2.1.2 Private Key

The ideal class group $cl(\mathcal{O})$ acts freely and transitively on $\mathcal{E}$. The element $\{\mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_k^{e_k}\}$ of $cl(\mathcal{O})$ is represented in CSIDH as the private exponent vector. This array of k elements $(e_1, \ldots, e_k)$ forms the private key. From now on, we will call a single element of the vector a key coefficient. Each key coefficient e_i is a random element in the range $[-m, m]$. m is a bound obtained from the parameter generation to store approximately $\frac{\log_2 p}{2}$ bits. The sign of the key coefficient describes the direction of the walk: Walking e steps from some point and then $-e$ steps results in returning to the starting point. This is a result of the dual isogeny theorem, which states that for each isogeny $E \to E'$, a corresponding isogeny $E' \to E$ exists. The dual isogeny can be directly used to invert the key: negating each key coefficient $e_i \mapsto -e_i$ results in the inversion of k, which we will denote as k^{-1}. Conversely, it is also possible to add two private keys, where their respective coefficient vectors are added, which we will denote as $k + l$, with k and l being CSIDH private keys. This will be important

from Section 2.4 onwards, as it gives rise to significant speedups in computing the group action.

Following the notation in [LGd21], we use $\mathbf{s} * E$ as shorthand to denote the class group action between $\mathfrak{s} = \{\mathfrak{l}_1^{s_1} \cdots \mathfrak{l}_k^{s_k}\}$ and E using the vector $\mathbf{s} = (s_1, \ldots, s_k)$.

2.1.3 Public Key.

The CSIDH public key is the Montgomery coefficient $A \in \mathbb{F}_p$ of the supersingular curve $E : y^2 = x^3 + Ax^2 + x$ and deterministically obtained by repeatedly applying the private key to the base curve $E_0 : y^2 = x^3 + 0 \cdot x^2 + x$. Of p possible public keys, $\sqrt{p}$ of those keys are valid, meaning that they describe supersingular curves.

2.1.4 Computational Problems.

For a clear security analysis, we now discuss the general key recovery problem in CSIDH and present a lemma that helps argue security throughout the paper. Problem 1 corresponds to Problem 10 in the original CSIDH paper. Lemma 1 directly follows from Problem 1.

Problem 1 (Key Recovery Problem) *Given the two different supersingular curves $E, E' \in \mathcal{E}$, find an $\mathbf{s} \in cl(\mathcal{O})$ such that $\mathbf{s} * E = E'$.*

[LGd21] give a useful lemma showing that sampling elements of the class group $Cl(\mathcal{O})$ is statistically close to uniform. We present a slightly shortened and modified version in Lemma 1, and will use it throughout the paper.

Lemma 1 (Computational Hiding in CSIDH) *Given a curve $E \in \mathcal{E}$ and a distribution D on $Cl(\mathcal{O})$, let $D * E$ be the distribution on $\mathcal{E}$ of $a * E$ for for $a \leftarrow D$. If D is statistically indistinguishable from the uniform distribution on $Cl(\mathcal{O})$, $D * E$ is statistically indistinguishable from the uniform distribution on $\mathcal{E}$. Therefore, we say that D statistically hides E.*

2.1.5 Parameterization and Security

The size of the prime p denotes the security parameter of CSIDH. There is an unusually heavy disagreement in the literature on the secure parameterization of CSIDH [BLMP19, BS20, Pei20]. Several theoretical and concrete quantum attacks with subexponential complexity dispute that a prime p which is 512 bits long is sufficient for security. Related work on OPRFs [BKW20] recommends using 2260-bit prime numbers for aggressive parameterization and 5280-bit primes for a conservative instantiation based on analysis of these algorithms. Recent work analyzing and implementing CSIDH with bigger primes concludes that a bitlength of at least 2048 bits, up to 9216 bits is necessary [CSCJR22].

For best comparability with other implementations, we use the 512-bit reference implementation of CSIDH throughout this paper, but point out that

the prime length may not be sufficient. An additional benefit of this implementation is the use of hardware instructions, which speed up the computation significantly.

2.2 CSI-FiSh

Building on CSIDH, the signature scheme CSI-FiSh introduces uniform representation in the class group elements. In their paper this is necessary for the Fiat-Shamir transformation, but the use-cases stretch beyond signatures. Intuitively, increasing the bound m of the key coefficient comes closer to sampling uniformly over the class group. To sample fully uniform keys, CSI-FiSh computes the class number and class group structure. From this, a relational lattice is generated, and the basis is reduced using Babai's nearest plane algorithm. Due to the different distribution of the class group ideals, the group action is around 15% slower.

2.3 Notation

We denote the sequential application of the group action $\mathsf{csidh}(\mathsf{csidh}(E, \mathbf{a}), \mathbf{b})$ as $\mathbf{b} * (\mathbf{a} * E)$. We denote the zero curve as E_0 and any other curve as E, potentially annotating it to give more context. For example, the result of applying some key $\mathbf{c}$ will be denoted $E_c = \mathsf{csidh}(\mathbf{c}, E_0) = \mathbf{c} * E_0$.

We will use an ideal functionality $\mathsf{keygen}()$ to sample random, uniform CSIDH private keys. $[\mathbf{k_1}, \mathbf{k_2}] \xleftarrow{\$} \mathsf{keygen}()$ samples two random, independent and uniform keys. Sampling several curves will sometimes be denoted as K. We will call a curve E *randomized* after sampling a private key $\mathbf{r} \xleftarrow{\$} \mathsf{keygen}()$ and computing $E' = \mathbf{r} * E$. We remove the property after applying $\mathbf{r}^{-1}$ to the curve E', therefore removing the randomness.

We write a vector $\mathbf{v}$ as a bold, lowercase variable, which is used for private exponent vectors. Adding the coefficients of two vectors is denoted as $\mathbf{a} + \mathbf{b}$, and coefficient-wise subtraction is denoted as $\mathbf{a} - \mathbf{b}$. From this point onwards, we denote the party holding the input X to the OPRF as the client and the party holding the key as the server, as this makes it more which entity holds what part of the OPRF input.

2.4 The Naor-Reingold Pseudorandom Function (NR-PRF)

The Naor-Reingold PRF [NR04] is a generic construction for PRFs from Abelian group actions. It is widely used in the literature and practice. The PRF requires $n + 1$ group elements, or keys, for n bits of PRF input. To compute the PRF, we take the initial group element k_0. For each input bit x_i for $i \in [1, n]$, a group action is performed if and only if the i^{th} bit x_i is set. The abstract functionality is outlined in Figure 1.

2.5 Oblivious Transfer and Naor-Reingold OPRF

The NR-PRF gives rise to oblivious evaluation using oblivious transfer (OT). In the simplest form, OT takes two messages m_0, m_1 from the sender, usually the server, and a choice bit c from the receiver, usually the client. The protocol functionality returns m_c to the client. The protocol is secure when the client learns nothing about m_{1-c} and the server learns nothing about c.
To use OT for an algebraic OPRF, the input $X = [x_1, \ldots, x_n]$ is bit-decomposed and used as the choice bit c. The server returns ${k_i}^c \circ r_i$. The client knits together the received elements via the group action, obtaining a blinded group element. To finalize the computation, the server sends evaluates the inverse of all blinding elements with the key and sends them to the client, who computes the group action with the finalization element and the blinded group element to obtain the NR-PRF.

3 Attacking and Repairing the Generic Naor-Reingold OPRF from CSIDH

Previous work [BKW20] describes the generic Naor-Reingold OPRF in CSIDH to compare against their SIDH-based proposal. While the latter has been broken [BKM+21] and subsequently repaired [Bas23], the approximations for the Naor-Reingold OPRF from CSIDH are widely cited in the literature and have not been studied further. We fill this gap with a thorough investigation in both the NR-PRF and the NR-OPRF from CSIDH.

3.1 Instantiating the NR-PRF from CSIDH

To instantiate the NR-PRF with CSIDH, the protocol samples $n + 1$ CSIDH private keys and computes the group action as in Figure 2. The textbook variant of the PRF outlined in Figure 2 is prohibitively slow, requiring $n + 1$ sequential group actions to compute the PRF for n input bits. The CSIDH-512 implementation takes around $20ms$ per group action when using Assembly instructions. A recent paper describes an effective way to evaluate the PRF by splitting the evaluation into two parts [ADMP20]: First, a subset-product, in the case of CSIDH addition of all key elements where $x_i = 1$, is computed. This first step can be parallelized. The group action is then evaluated in a second step on the base curve.

$$F_{NR}\left((k_0, k_1, \ldots, k_n, E_0), (x_1, \ldots, x_n)\right) := k_0 \circ k_1^{x_1} \circ \ldots \circ k_n^{x_n}$$

Figure 1: The Naor-Reingold PRF from a generic group action, denoted $\circ$. Classic examples for $\circ$ are modular multiplication or point addition on elliptic curves. The exponentiation with x_i may be read as *perform* $\circ$ *if input bit is set.*

$$F_{NR-CSIDH}\left((\mathbf{k_0}, \mathbf{k_1}, \ldots, \mathbf{k_n}), (x_1, \ldots, x_n)\right) := \mathbf{k_0} * \mathbf{k_1}^{x_1} * \ldots * \mathbf{k_n}^{x_n} * E_0$$

Figure 2: Naor-Reingold PRF from CSIDH using E_0 as a starting curve.

$$F_{NR-CSIDH-OPT}\left((\mathbf{k_0}, \mathbf{k_1}, \ldots, \mathbf{k_n}, E_0), (x_1, \ldots, x_n)\right) := \left(\mathbf{k_0} + \sum_{i=1}^{n} \mathbf{k_i}^{x_i}\right) * E_0$$

Figure 3: Optimized two-step Naor-Reingold PRF from CSIDH. The first step is a subset-sum of the required keys and the second step is the application of the group action to the base curve E_0.

The subset-sum computation requires a tiny tweak in the CSIDH implementation[1], from 8-bit to 32-bit key elements to avoid overflows. Other than adding addition and subtraction subroutines, the implementation is the same. In Figure 4, we benchmark the PRF computation for input sizes between 1 and 512 bits. We see that the two-step computation approach significantly reduces the evaluation time. The optimization also saves $n-1$ computations of the first step of the algorithm, which is computing a point of the correct order. This step is more expensive the smaller the value of ℓ_i, which is particularly nice for an aggressive parameter choice in CSIDH, as the probability of sampling a correct point is $\frac{\ell_i - 1}{\ell_i}$.

A nice property of this PRF is that it is updatable; that is, if parts of the input change, updating the output requires a single group action to update the PRF. This is useful for applications requiring to hash multiple inputs so the individual inputs differ in less than $\frac{n}{2}$ bits. In Figure 5, we show that the effort between recomputing the OPRF and updating a previous result holds fairly clearly to our expectations: It is cheaper to recompute the OPRF when less than 128 bits differ, if it is more, recomputation is more efficient. Note that the divergence is due to the non-uniform keys in CSIDH.

3.2 Oblivious NR-PRF from CSIDH

Following Protocol 24 in [BKW20] for a NR-OPRF from CSIDH, the parties engage in n $\binom{2}{1}$-OTs, where the key is blinded with a random group element. Using our trick from Section 2.4, a correct intuition is to instantiate the OT with $(\mathbf{r_i}, \mathbf{k_i} + \mathbf{r_i})$, finalizing the OT by sending $\mathbf{k_0} * \sum_{i=1}^{n} -\mathbf{r_i}$. We sketch this protocol in Figure 6.

[1] All CSIDH benchmarks use the reference implementation from `https://yx7.cc/code/csidh/csidh-latest.tar.xz`

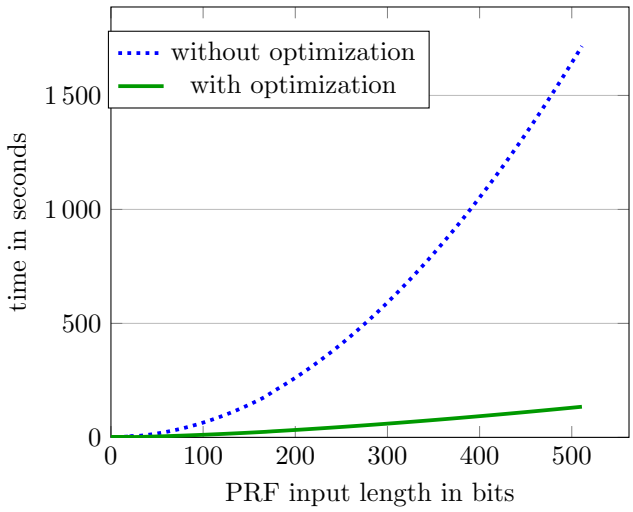

Figure 4: Runtime divergence between the traditional Naor-Reingold CSIDH PRF in blue and the same PRF with our optimization in green for different bit lengths.

3.3 Attacking the Construction

While producing a correct result, this OPRF leaks the server key in a passive attack as the key is not uniformly random. [2] A passive, semi-honest adversary can observe the distribution of the blinded keys by always choosing $c = 1$. Consider blinding a key element $k_i = m = 5$ with a random element that only needs a few invocations to see that the blinded key is always within the range $[0, 10]$. Over several iterations, this leaks the entire key, parts of it even at the first try, as a blinded 10 and -10 immediately show that both the key and the blinding element were 5 or -5, respectively. Some tests, where a static key is repeatedly blinded, shows that after approximately 65 evaluations, the entire key would be leaked. This does not take into account possible sophisticated methods such as guessing for even more restricted settings as parts of the key are immediately clear and, if only a few bits are missing, brute force is possible. We provide an implementation of this simulation.

[2] In personal communication, authors of [BKW20] confirmed that their specific instantiation of their construction using class groups (or isogenies) blinds the class group element representing the key by multiplying a random element, but that the non-uniform key distribution leads to the CSIDH instantiation of protocol [BKW20] being "currently broken".

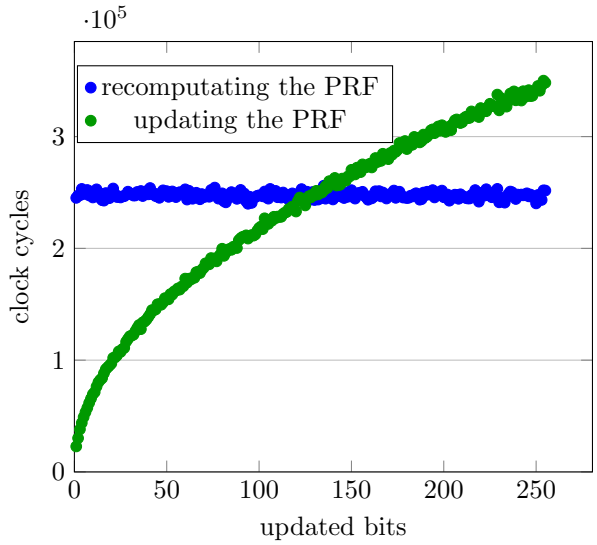

Figure 5: Runtime divergence between updating x bits of the PRF vs. recomputing the full 256 bits of the PRF, averaged over 100 runs.

3.4 Fixing the NR-OPRF

The non-uniform key distribution has already been discussed in the context of a Σ-commitment scheme from CSIDH. The first solution in the signature scheme SeaSign [DG19] is rejection sampling, concretely the Fiat-Shamir transformation with aborts. This technique was initially proposed by Lyubashevsky [Lyu09] to prevent leakage of the private key in lattice signatures.

To translate the Fiat-Shamir technique with aborts to the CSIDH setting, SeaSign uses somewhat short, long-term secret keys s with coefficients $s_i \in [-B, B]^k$ and large, ephemeral secret keys r with each coefficient $r_i \in [-(\delta + 1)B, (\delta + 1)B]^k$, rejecting any r where $r - s$ if any coefficient is outside of the range $[-(\delta)B, (\delta)B]$. For the NR-OPRF, the long-term sender keys are then obviously the short keys s and the ephemeral keys are sampled as r.

While using tactics from SeaSign is a good mitigation, it puts a significant computational load on the server and introduces the drawbacks of lattice signatures in the scheme. Additionally, the large ephemeral keys add significant communication overhead to the protocol.

Instead of rejection sampling, directly sampling a uniform key would significantly improve the protocol. Therefore, we propose to mitigate the leakage by using the sampling algorithm from the signature scheme CSI-FiSh [BKV19]. The protocol in Figure 6 would largely remain the same, with $\mathbf{k_i} + \mathbf{r_i}$ being a reduced element of the class group. We give a sketch of the server and client

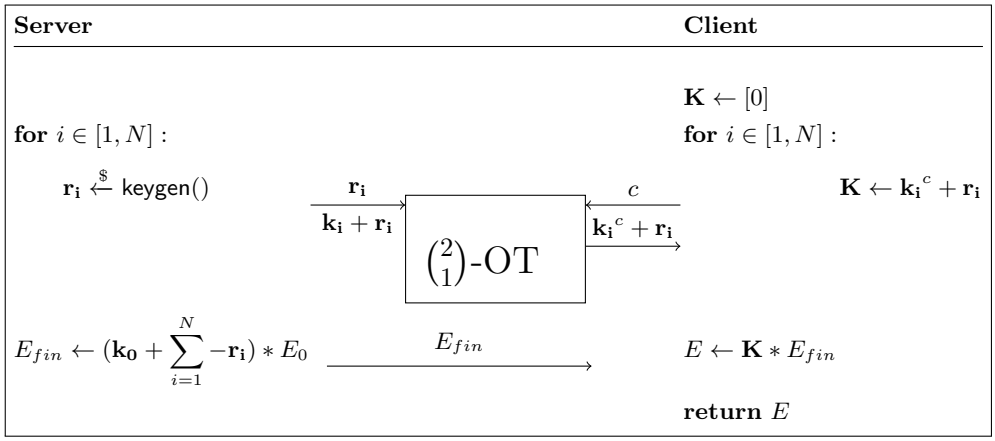

Figure 6: Evaluating the NR-OPRF with CSIDH and N OT calls.

protocols in Figure 7 for Clarity. Note that the curve E_t has an unknown endomorphism ring.

3.4.1 Performance

The main drawback of this mitigation is the computational complexity, as the keys are reduced using lattice computation and the overall group action is about 15% slower than the original CSIDH group action. An advantage is that the sender only has to precompute the class structure once, and after that the computational load on the client, aside from the OT, is relatively low.

3.4.2 Trusted Setup

The main disadvantage of this NR-OPRF is that the underlying OT protocol based on isogenies [LGd21] requires a supersingular curve with an unknown endomorphism ring, also called SECUER. Currently, there are no known efficient algorithms to construct such a curve. A recent paper [BCC+23] proposes generation for supersingular curves over $\mathbb{F}_{p^2}$ used in SIKE. For CSIDH's $\mathbb{F}_p$, the authors [BCC+23] mention that their techniques are not well suited, and note it as an open problem. Therefore, using the OPRF protocol either requires a different OT protocol without a trusted setup or an efficient construction of SECUERs over $\mathbb{F}_p$.

Alternate constructions of OT from CSIDH have similar problems: The semi-honest protocol of [dSGOPS20] gives similar performance to the OT protocol of [LGd21] while requiring two trusted curves for the setup. A good alternative may be the single-bit OT of [ADMP20]. It requires a distribution closer to uniform than CSIDH, however, it may be an attractive alternate choice. A

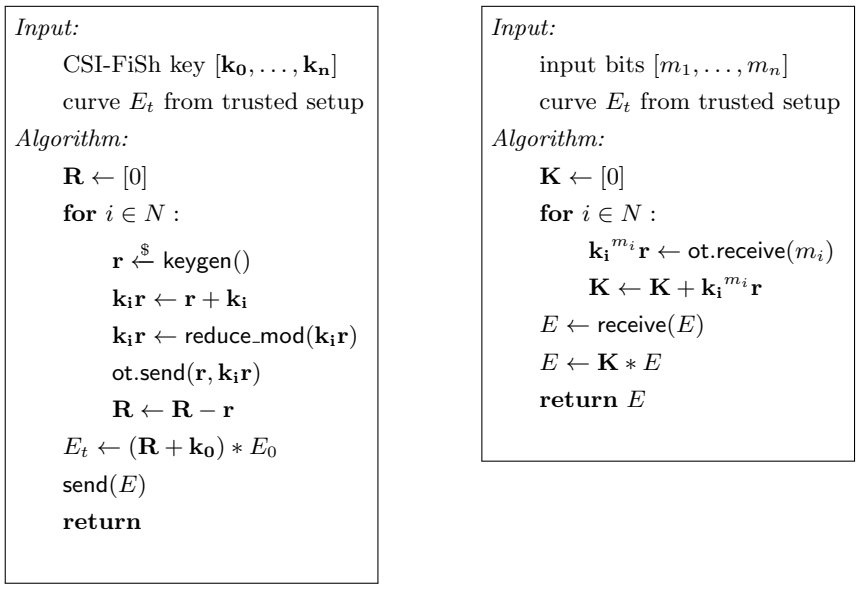

Figure 7: Server and Client algorithm for the NR-OT OPRF.

significant drawback of this protocol are the number of isogeny comutations depending on γ. An efficient protocol may use similar techniques as [KRS+19] to minimize the number of necessary group actions.

4 OPUS: Oblivious Pseudorandom Function using CSIDH

We observe that the blinding of the class group elements hampers the security of the scheme significantly: it introduces new security assumptions to the scheme and makes the protocol less efficient. Instead of sending private keys over the network, we propose OPUS, a novel construction that only sends evaluated curves, that is, CSIDH public keys. In the protocol, both parties iteratively blind their intermediate results, with the client getting anything useful only in the end, beforehand computing over randomized curves. This eliminates the need for a trusted setup, which is the main obstacle hampering other OPRF protocols from CSIDH. The main operations in OPUS are blinding and key addition. In each step, the client blinds a curve, starting with E_0, with a random class group element $\mathbf{r_{c,i}}$ and sends it to the server, which returns the curve blinded again with its own, fresh blinding element $r_{s,i}$ and once with the own blinding element and the key. Now, the client decides based on the i^{th} bit of the input with which curve the computation should continue, blinding again to ensure the server learns nothing about their choice. By the hiding lemma 1, this

perfectly protects the client input and the server keys from malicious parties. We outline our novel OPRF in Figure 8.

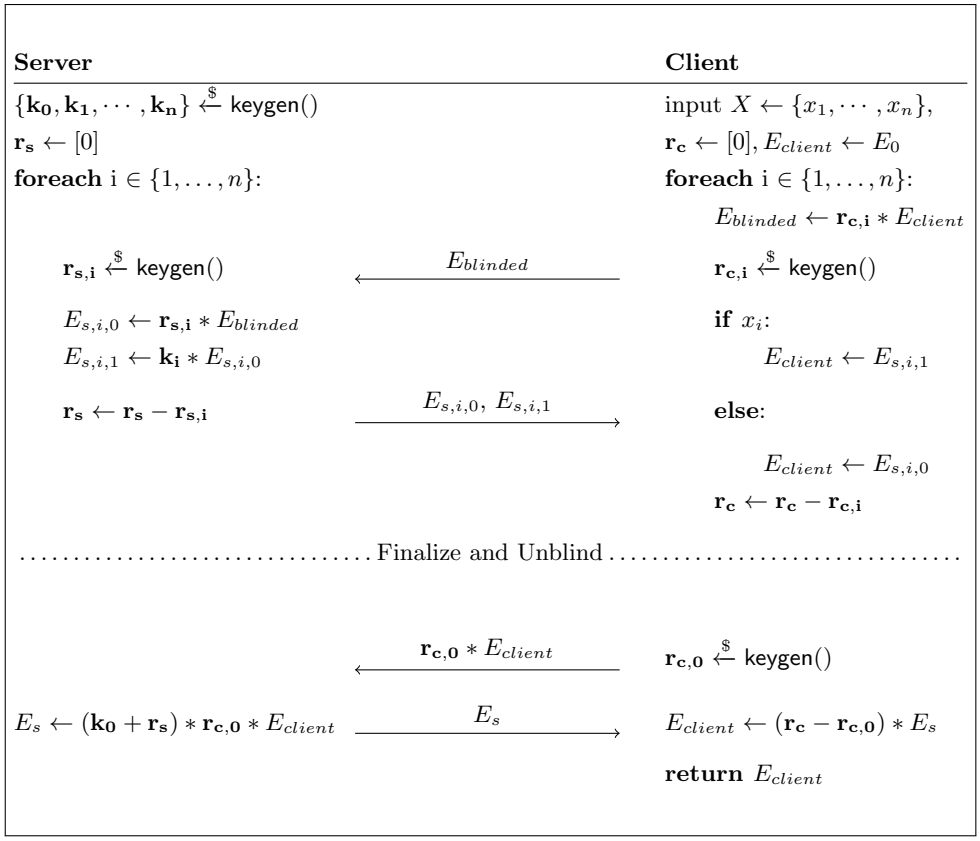

Figure 8: The full protocol of our novel OPRF *OPUS*.

4.1 Efficiency

We use a similar trick as in Section 2.4 to speed up the computation, as we aggregate the blinding keys in an element R. This reduces the necessary group actions to $3n + 2$ for the OPRF computation, with $2n + 1$ group action computations for the server and $n + 1$ for the client. Experimental runtimes can be found in Table 1, and concrete runtimes in Table 2. A nice part of OPUS is that the server carries the highest computational load, while the client only has to perform $n + 1$ CSIDH computations.

To concretize the overhead imposed by the rounds, we rented virtual machines all over the globe and used them as clients performing OPUS with a

Table 1: Comparison between PRF and OPRF execution time on an Intel i5-10210U with 16 GB RAM and Ubuntu L22.04 locally, averaged over 100 runs.

Bit-length	Keygen Keygen	PRF	OPRF Overall	OPRF User	OPRF Sender
128	0.14ms	183.98ms	9.54s	3.29s	6.25s
256	0.27ms	243.55ms	18.32s	6.24s	12.08s
512	0.54ms	362.16ms	37.06s	12.54s	24.52s

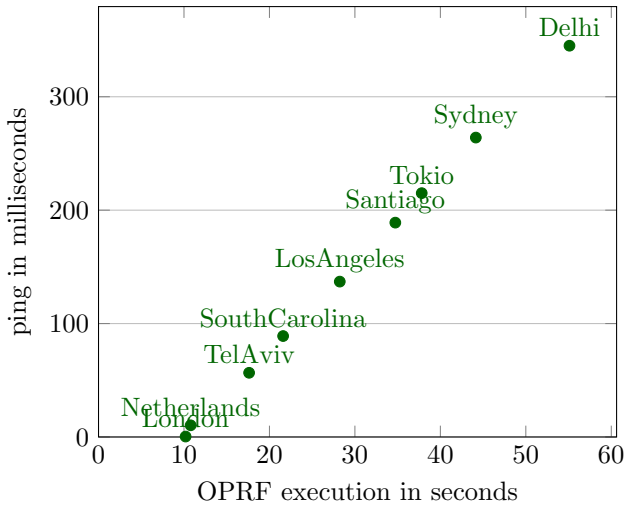

Figure 9: Online runtimes of different clients computing OPUS with a bitlength of 128 with a server in London. All machines run on Debian 11 using the simplest Google Cloud instance.

server in London. As clear from Figure 9, the runtime of OPUS directly corresponds to the round-trip time of the ping. In a real-life setting, this overhead may be mitigated by running several, distributed instances of a server.

4.2 Possible Extensions of OPUS

We now give intuitions for possible extensions of OPUS. Specifically, we discuss how to potentially lift OPUS to a distributed or threshold OPRF, batch and update the OPRF, and give other notions that may be addressed in future work, either by refuting our intuitions or using them. Several notions were discussed in a recent SoK [CHL22].

Distributed and threshold OPRF To increase resilience against key compromise, the server key K can be distributed across several servers. Threshold OPRFs distribute the keys such that only t out of n honest servers are required to produce an OPRF result. A common approach is distributing the key using Shamir's secret sharing [Sha79]. For CSIDH, a recent paper [DM20] demonstrates threshold key sharing. Their results should be directly applicable to OPUS, at the penalty of requiring t more rounds.

Batched OPRF While there is no direct way to batch OPUS, evaluating several OPRFs in one session can be used to amortize the overhead imposed by the rounds, as sending several blinded curves for the OPRF increases the traffic, but not the round complexity. Obviously, the blinding keys must be uniformly random.

Updatable OPRF Only updating a part of the OPRF seems trickier. In a simple version, the client sends the indices where two inputs X_1, X_2 differ. The parties then engage in a reduced execution of OPUS, where the server responds with $(\mathbf{r} * \mathbf{k}^{-1} * E, \mathbf{k} * \mathbf{r} * E)$ for the given indices.

As the server knows the index of where two evaluations differ, another version of this is to send some dummy indices and require the server to respond with $(\mathbf{r}*\mathbf{k}^{-1}*E, \mathbf{r}*E, \mathbf{k}*\mathbf{r}*E)$, with $\mathbf{r}*E$ being consecutively used if the index was a dummy index. This approach may reduce the latency introduced by the rounds as e.g. visible in Figure 9 and may be attractive for settings where several OPRF evaluations are necessary, such as private set intersection [KRS+19].

Strong OPRF An OPRF is *strong* [FIPR05] when the client learns nothing about the key from the result. If the client learns anything about the key, they break Problem 1. Therefore, OPUS is a strong OPRF.

Partial Obliviousness Partially Oblivious Pseudorandom Functions (POPRFs) reveal some client input to the server. This can be useful in many settings e.g. to provide some identification marker. While OPUS does not directly provide the functionality, this may work with an identification scheme, e.g. from CSI-FiSh.

Committed in- and outputs Protocols using the OPRF as a building block may require that the client commits to an input or the server obtains a commitment about the output [CHL22]. Both require some algebraic structure. While we believe this is possible with zero-knowledge proofs, our current construction of OPUS does not allow commitments. There is currently no committed NR-based OPRF [CHL22].

Key Rotation and Proactive Security Refreshing keys is necessary to recover from a compromise, also called key rotation. OPUS is currently unable to perform key rotation.

5 Security Analysis

To prove our novel OPRF secure in the standard model, we will show correctness, hiding, binding and one-more security, giving reductions to either the key recovery problem in Problem 1 or the hiding lemma in Lemma 1.

5.1 Correctness

We now show that the protocol OPUS in Figure 8 generates output in correspondence to the CSIDH NR-PRF F_{NR} from Figure 1.

Proposition 1 (OPUS produces correct NR-PRF outputs) *For all keys $\mathbf{k} \in \mathcal{K}$ and inputs $\mathbf{x} \in \{0,1\}^n$, the output of an honest computation of OPUS is an evaluation of F_{NR}. That is $\mathbb{P}[F_{OPUS}(\mathbf{k}, \mathbf{x}) = F_{NR}(\mathbf{k}, \mathbf{x})] = 1$, with the probability being over the internal randomness of OPUS.*

Proof 1 (Proof: Correctness of OPUS) *Given input $X = (x_1, \ldots, x_n)$ and keys $K = (\mathbf{k_0}, \ldots, \mathbf{k_n})$, the client $\mathcal{C}$ generates a random key $\mathbf{r_{c,i}}$ for each $i \in [1, n]$ and initializes $E \leftarrow E_0$. For each $i \in [1, n]$, $\mathcal{C}$ sends a randomized curve $\mathbf{r_{c,i}} * E$ to the server $\mathcal{S}$, who samples $\mathbf{r_{s,i}}$ and returns $E'_0 \leftarrow \mathbf{r_{s,i}} * E$ and $E'_1 \leftarrow \mathbf{k_i} * \mathbf{r_{s,i}} * E$ to $\mathcal{C}$. If $x_i = 1$, $\mathcal{C}$ sets $E \leftarrow \mathbf{k_i} * \mathbf{r_{s,i}} * E$ and $E \leftarrow \mathbf{r_{s,i}} * E$ otherwise. Clearly, repeating this step n times is equivalent to computing*

$$\left(\left(\sum_{i=1}^{n} \mathbf{r_{s,i}} + \sum_{i=1}^{n} \mathbf{r_{c,i}} + \sum_{i=1}^{n} \mathbf{k_i}^{x_i} \right) * E_0 \right).$$

The computation is finalized by $\mathcal{C}$ blinding the result again with the term $\mathbf{r_{c,0}}$ and sending it to the server, which applies $\mathbf{k_0}$ as well as the sum of the inverse blinding terms $\mathbf{r_s}$ such that

$$(\mathbf{k_0} - \sum_{i=1}^{n} \mathbf{r_{s,i}}) * \left(\left(\mathbf{r_{c,0}} + \sum_{i=1}^{n} \mathbf{r_{s,i}} + \sum_{i=1}^{n} \mathbf{r_{c,i}} + \sum_{i=1}^{n} \mathbf{k_i}^{x_i} \right) * E_0 \right),$$

which is equivalent to

$$\left(\sum_{i=0}^{n} \mathbf{r_{c,i}} + \mathbf{k_0} + \sum_{i=1}^{n} \mathbf{k_i}^{x_i} \right) * E_0.$$

The client is left to compute the inverse of their respective blinding elements such that

$$\sum_{i=0}^{n} -(\mathbf{r_{c,i}}) * \left(\sum_{i=0}^{n} \mathbf{r_{c,i}} + k_0 + \sum_{i=1}^{n} \mathbf{k_i}^{x_i} \right) * E_0,$$

which is equivalent to computing

$$\left(\mathbf{k_0} + \sum_{i=1} \mathbf{k_i}^{x_i} \right) * E_0.$$

Therefore, OPUS correctly evaluates the NR-PRF for honest parties.

5.2 Interlude: Freely and Transitively Acting Class Group

Before proving hiding and binding, we revisit the necessary properties of the class group action for our security proves. Given a group element $g \in G$ and two elements $x, y \in cl(\mathcal{O})$, the group action is said to act *freely* if,

$$\forall (x, y) \in cl(\mathcal{O})^2 \exists g \in G : gx = y.$$

Further, the class group action is said to act *transitively*, if

$$\forall x \in cl(\mathcal{O}) : gx = x \implies g = I,$$

with I being the identity element of the group. This is important to capture all isomorphic curves. A group action that acts both freely and transitively has that

$$\forall (x, y) \in cl(\mathcal{O})^2 \exists! g \in G : gx = y.$$

CSIDH samples ideal classes within $[-m, m]$. Increasing m leads to distributions closer to uniform. CSIDH only offers statistic indistinguishability from uniform, as discussed in[LGd21]. For uniform sampling, CSI-FiSh[BKV19] can be used.

Proposition 2 (Randomized Transcript) *We propose that the transcript of OPUS is randomized, that is, any PPT adversary $\mathcal{A}$ does not learn about the input or the key from the transcript with more than negligable probability.*

Proof 2 *Due to the hiding Lemma 1, the intermediate messages do not give any information about the underlying curve as the randomized curve is statistically indistinguishable from a random curve over $\mathcal{E}$. Therefore, the transcript is statistically indistinguishable from a random transcript and therefore does not give information to a passive PPT adversary with more than negligable probability.*

5.3 Hiding

For our OPRF to be oblivious, the client input and the server keys must be hidden both from the other party and outside observers, that is, informally, any information obtained from an evaluation by an adversary should be as likely to have been seen regardless of the underlying secret value.

Proposition 3 (Client-Hiding) *For all $K \in \mathcal{K}$ and $X \in \{0,1\}^n$, let OPUS be executed with either X or a uniformly random $X' \in \{0,1\}^n$ as client input. The probability distributions of the server-side transcript of OPUS in the two different cases are statistically indistinguishable.*

Proof 3 *Let K be an arbitrary key and consider each element $x_i \in X$. For $i = 1$, the server learns nothing about the client input as there is no dependence, and $\mathbf{r_{c,0}} * E_0$ does not give any information. In each step $i > 1$, the server receives a message dependent x_i which is either $\mathbf{r_{c,i}} * E_i$ or $\mathbf{r_{c,i}} * E_j$, where E_i, E_j are known to the server and $\mathbf{r_{c,i}}$ is a random class group element. As the CSIDH class group action acts freely and transitively, any $E \in \mathcal{E}$ can be*

mapped to any other by exactly one group element. By Lemma 1, $\mathbf{r_{c,i}}$ *statistically hides the curve* E_{client} *in the transcript, regardless of the value of* E_{client}*. Thus each individual message of the client-side transcripts for inputs* X *and* X *have distributions that are statistically indistinguishable for both the server and a passive interceptor of the entire transcript.*

Proposition 4 (Server-Hiding 1) *Given the randomized server messages* $\{E_{s,i,0} \leftarrow \mathbf{r_{s,i}} * E_{blinded}, E_{s,i,1} \leftarrow \mathbf{r_{s,i}} * \mathbf{k_i} * E_{s,i,0}\}_{i\in[1,n]}$*, no PPT adversary* $\mathcal{A}$ *controlling the client can learn the partial evaluations* $\{E_i = (\sum_{j=1}^{i} \mathbf{k_j}) * E_0\}_{i\in[1,n]}$ *with more than negligible success probability. This holds for all keys* $\mathbf{k} \in \mathcal{K}$ *and inputs* $X \in \{0,1\}^n$*. That is,*

$$\mathbb{P}[\{\hat{E}_i\}_{i\in[1,n]} \leftarrow \mathcal{A}(\{E_{s,i,0}, E_{s,i,1}\}_{i\in[1,n]}, X) :$$

$$\{\hat{E}_i\}_{i\in[1,n]} = \{E_i\}_{i\in[1,n]}] \leq \epsilon(n),$$

where the probability is over the internal randomness of $\mathcal{A}$ *and OPUS.*

Proof 4 *The proposition follows from Problem 1 and the hiding Lemma 1. For each* i*, both* $E_{s,i,0}$ *and* $E_{s,i,1}$ *consist of the group action performed between a key that the client does not know and a curve that the client does know. The hiding lemma guarantees us that knowing* $E_{s,i,1}$ *does not help in trying to learn* $\mathbf{k_i} * E_{blinded}$*, since* $\mathbf{r_{s,i}}$ *is statistically indistinguishable from uniform random. The hardness of Problem 1 guarantees us that no PPT adversary can derive* $\mathbf{k_i}$ *from knowing* $E_{s,i,1} \leftarrow \mathbf{k_i} * E_{s,i,0}$ *and* $E_{s,i,0}$*. Therefore, no PPT adversary can with non-negligible probability derive the partial evaluation from either one of* $E_{s,i,0}$ *and* $E_{s,i,1}$*, or the difference between them. Since this holds for each iteration separately, it also holds for them as a whole due to the independence between the randomness in each round.*

Proposition 5 (Server-Hiding 2) *For all* $K \in \mathcal{K}$*, no PPT adversary is controlling the client that, with more than negligible probability, can derive any knowledge about* K *from the server messages* $\{E_{s,i,0} \leftarrow \mathbf{r_{s,i}} * E_{blinded}, E_{s,i,1} \leftarrow \mathbf{k_i} * E_{s,i,0}\}_{i\in[1,n]}$*. That is,* $\forall K \in \mathcal{K}$*,*

$$\mathbb{P}[\hat{K} := \{\hat{\mathbf{k}}_\mathbf{0}, \ldots, \hat{\mathbf{k}}_\mathbf{n}\} \leftarrow \mathcal{A}(\{E_{s,i,0}, E_{s,i,1}\}_{i\in[1,n]}, X) : \hat{K} = K] \leq \epsilon(n),$$

where the probability is over the internal randomness of $\mathcal{A}$ *and OPUS.*

Proof 5 *The proposition follows directly from the assumed hardness of Problem 1, due to the independence between the randomness in each iteration.*

Due to the non-uniform distribution of CSIDH keys, OPUS only statistically hides the input and key. If unconditional hiding is necessary, class group elements have to be computed as in CSIDH to obtain uniform keys.

5.4 Binding

A secure PRF needs to be collision resistant, that is, it should be hard to find X_1, X_2 such that $F_K(X_1) = F_K(X_2)$. For Naor-Reingold PRFs, some colliding inputs are trivially possible if certain keys of $K = (\mathbf{k_0}, \ldots, \mathbf{k_n})$ are *weak*. This is the case if K contains the zero key, repeating keys, keys that are the inverse of another, or additive combinations of keys that map to a distinct subsets of other keys.

Example 1 (Trivial Collisions) *If all* $\mathbf{k_i} = [0]^k$ *for* $i > 0$*, then all possible client inputs will lead to a collision, and therefore finding such a collision is trivial.*

Let $K = (\mathbf{k_0}, \ldots, \mathbf{k_n})$*. For* $n = 1$*, if* $\mathbf{k_1}$ *is the zero-key and* $\mathbf{k_0}$ *arbitrary, both* $X = [0]$ *and* $X' = [1]$ *produce the same output. If not all round keys are zero but only most of them, we still consider the collisions to be trivially easy to find, in the sense of them occurring with overwhelming probability when one selects client inputs at random.*

The probability of randomly drawing the zero key is $\frac{1}{(2m+1)^k}$, which is small even for CSIDH-512. Additionally, it is trivial to check for this case during key generation.

Example 2 (Combinatorial Collisons) *For the case with* $n = 2$*, in addition to trivial collisions, now there are two more possible collisions.*

1. *For* $n > 1$*, there is a risk of some* $\mathbf{k_1}, \mathbf{k_2}$ *that are equal. Take* $\mathbf{k_1} = \mathbf{k_2}$*, this produces a collision between* $X = [0, 1]$ *and* $X' = [1, 0]$*.*

2. *For* $n > 1$*, there is a risk of some* $\mathbf{k_1}, \mathbf{k_2}$ *that are inverses of each other, that is,* $\mathbf{k_1} - \mathbf{k_2} = [0]^k$*. Then,* $X = [0, 0]$ *and* $X' = [1, 1]$ *collide.*

For $n = 2$ the probability of drawing keys that are the same are or inverses are each again $\frac{1}{(2m+1)^k}$. Motivated by Example 2, we now show how there are non-trivial collisions for up to n keys.

Example 3 (Subset Collisions) *Let* $K = (\mathbf{k_0}, \mathbf{k_1}, \ldots, \mathbf{k_n})$ *with* $\mathbf{k_0}$ *being arbitrary. Collisions may occur if a subset of keys* $\mathbf{k_i}, \ldots, \mathbf{k_j}$ *adds to zero or another subset. For example, with* $\mathbf{k_1} + \mathbf{k_2} = \mathbf{k_3}$*, then* $X = [1, 1, 0]$ *and* $X' = [0, 0, 1]$ *collide. Similarily,* $\mathbf{k_1} + \mathbf{k_2} = \mathbf{k_3} + \mathbf{k_4}$ *collide on* $X = [0, 0, 1, 1]$ *and* $X' = [1, 1, 0, 0]$*, and so on.*

By continuing the examples to larger n it does seem like that, while the probability of collision-prone round keys exist, the probability of drawing two client inputs that collide decreases as the length of tuples increase. The reason is that as each new round key is added, there is not only the added probability that it is the zero-key on its own, but also the added probability that it creates a subset sum within the key set as showin in Example 3. Such a collision requires two

client inputs X, X' to differ exactly so the subset-sum is found. The probability for this is exponentially small in the length n. Therefore, the probability of drawing a collision-prone key set in CSIDH seems to be a small factor dependent on n, which we will denote as ν, in the form $\frac{\nu}{(2m+1)^k} = \frac{\nu}{2^{k \log_2(2m+1)}}$. Therefore it seems that the probability of drawing collision-prone keys is negligible in $k \log_2(2m+1)$ and therefore also in $k \log_2(m)$.

Let us call a key set for which the probability of uniformly guessing two client inputs (without replacement) such that they collide is non-negligible in n *collision-likely*. Based on the intuition presented above, we conjecture that that collision-likely key sets are rare, more precisely:

Conjecture 1 (Rarity of collision-likely Key Sets) *The probability of drawing a key set K, containing $n+1$ keys, such that K is collision-likely is negligible in $k \log_2 m$.*

Using Conjecture 1, we propose that OPUS is weakly binding.

Proposition 6 (OPUS is weakly binding) *Assume the hardness of Problem 1 and that Conjecture 1 holds. Then there is no PPT adversary controlling the client that, for a randomly sampled $\mathbf{k} \in \mathcal{K}$, can find a collision in OPUS for the client input with more than negligible probability. That is, for all PPT adversaries $\mathcal{A}$, with query access to $\mathsf{F}_{OPUS}(\mathbf{k}, \cdot)$ and the view of the client,*

$$\mathbb{P}[X, X' \leftarrow \mathcal{A} : \mathsf{F}_{OPUS}(K, X) =$$

$$\mathsf{F}_{OPUS}(K, X'), X \neq X' | K \xleftarrow{\$} \mathsf{keygen}()] \leq \epsilon(n).$$

The probability is over the internal randomness of $\mathcal{A}$, and the randomness in the key generation.

Proof 6 *The simplest collision, a preimage, where $\mathcal{A}$ uses an evaluation $E \leftarrow \mathsf{F}_{OPUS}(K, X)$ to derive knowledge about K from E, is contradicted by Problem 1. Picking random inputs X, X' is infesible given the size of $End_p(E) \approx \sqrt{p}$. Finally, finding subset-sum collisions is hard by Conjecture 1, as there is iat most a negligible probability (in $k, \log_2 m$) of K being such that collisions can be found by uniform guessing with non-negligible probability in n.*

5.5 OPUS is one-more unforgeable

To finally obtain a secure OPRF, we now prove one-more unforgeability, that is, asking for r instances of the OPRF does not enable a PPT client to compute a new, $r + 1^{th}$ evaluation without executing OPUS again.

Proposition 7 (One-More unforgeability) *Assume the hardness of Problems 1. Then given a uniformly sampled key $K \in \mathcal{K}$ and r input-output pairs $((X_1, \mathsf{F}_{OPUS}(K, X_1)), \ldots, (X_r, \mathsf{F}_{OPUS}(K, X_r)))$, with $X_i \in \{0,1\}^n$ freely chosen by a PPT adversary $\mathcal{A}$. There is no such $\mathcal{A}$, for which K is unknown,*

that can produce a valid new pair $(X_{r+1}, \mathsf{F}_{OPUS}(K, X_{r+1}))$ *with non-negligible probability, that is,*

$$\mathbb{P}((X_{r+1}, \hat{E}) \leftarrow \mathcal{A}((X_1, \mathsf{F}_{OPUS}(K, X_1)), \ldots, (X_r, \mathsf{F}_{OPUS}(K, X_r))) : \\ \hat{E} = \mathsf{F}_{OPUS}(K, X_{r+1}) | K \xleftarrow{\$} \mathsf{keygen}()) \leq \epsilon(n).$$

Proof 7 *Assuming both binding and randomized transcript hold,* $\mathcal{A}$ *needs to finding a valid combination of two OPRF results* $E_1 \leftarrow F_{OPUS}(K, X_1)$ *and* $E_2 \leftarrow F_{OPUS}(K, X_2)$*. Doing so would require the adversary to recover the path between the two curves, contradicting Problem 1. Therefore, even if* $\mathcal{A}$ *has access to all possible evaluations except one, the final one evaluation could still be any possible curve, and the adversary has no way of figuring out which one with non-negligible probability.*

5.6 Security Model

Due to the above computations, we claim that OPUS is a semi-honest OPRF. Due to all communication being realized over randomized curves we cannot detect if a participant in the protocol behaves maliciously. However, this gives us the guarantee of *privacy* against malicious users, that is, assuming Problem 1 and Lemma 1 hold, a malicious server learns nothing about the clients input and a malicious client learns nothing about the server keys.

6 Performance

To compare the different protocols for computing the Naor-Reingold OPRF from isogenies, we give a cost overview in Table 2. To approximate communication cost, we consider the number of bits sent as functions of the security parameters. To approximate computation costs, we consider the most expensive operation, the CSIDH group operation. only the total number of class group actions performed, again as functions of the security parameters, and assume all other operations are negligible in relation to these. For conciseness, let us denote $\log_2 p$ as σ and let γ be the security parameter (which, in practice, is equal to n). We point out that the communication cost estimates differ from the original estimate for the Naor-Reingold OPRF using [BKW20]. This is in part due to an update of the OT protocols used for NR-OT in [LGd21] and partly due to a small miscalculation in [BKW20], where they give communication complexity as

$$\gamma \cdot (3\sigma + 4\frac{\sigma}{2} + \gamma) = 5\sigma \cdot \gamma + \gamma^2, \tag{1}$$

where the $4\frac{\sigma}{2}$ term is for the *encryptions of class-group elements*, which are private keys of size $\frac{\log_2 p}{2}$, see Section 2.1.2. The encryptions in question are not of just class-group elements but these are also appended by a γ-bit random string, and the final round of the protocol also incurs the sending of an additional elliptic curve. Therefore the communication complexity of the NR-OT protocol is rather

$$\gamma \cdot (3\sigma + 4(\frac{\sigma}{2} + \gamma) + \gamma) + \sigma = 5\sigma \cdot \gamma + 5\gamma^2 + \sigma. \quad (2)$$

The comparison (Table 2) between the NR-OT protocol from Section 2.4 and OPUS from Section 4 shows that our novel proposal is more efficient in terms of computation (and thus also time) and falls probably somewhere in between the semi-honest and one-sided maliciously secure versions of NR-OT in terms of communication (dependent on parameter choices).

Table 2: Cost of the oblivious evaluation of the Naor-Reingold OPRF. σ denotes the CSIDH/CSI-FiSh security parameter and γ the security parameter. ◐ denotes a semi-honest party, and ● a malicious party.

work work	rounds rounds	comm. cost	isog. comp.	model (C-S)
NR-OT	2	$2\sigma \cdot \gamma + 2\gamma^2 + \sigma$	$5\gamma + 2$	◐-◐
NR-OT	4	$5\sigma \cdot \gamma + 5\gamma^2 + \sigma$	$11\gamma + 2$	●-◐
OPUS	$2\gamma + 2$	$3\sigma \cdot \gamma + 2\sigma$	$3\gamma + 3$	◐-◐

In Table 3, we show how the communication complexity compares for different parameters regimes to better illustrate the concrete complexity. From the figure it is clear that for reasonable parameter regimes such as these, OPUS is currently the most competition efficient OPRF that is secure against semi-honest adversaries in a post-quantum setting.

7 Related Work

OPUS and the generic NR-OPRF from isogenies are only two of several, recent proposals in post-quantum cryptography. Note that the estimates for the communication complexity may change drastically as the concrete security of CSIDH remains an open research question, see Section 2.1.5. We give further estimates in Table 3 and provide parameter-agnostic approximations in Table 2, showing that OPUS is still competitive even with larger parameters.

A recent proposal [Bas23] repairs the SIDH-based OPRF [BKW20] and also enables verifiability, that is, that the client can verify that the server used a certain key K. OPUS lacks this property. A drawback of the SIDH-based construction is that a trusted setup is necessary as well, which is expensive but possible over $\mathbb{F}_{p^2}$ [BCC+23].

On the lattice side, an initial proposal for round-optimal, verifiable OPRFs [ADDS21] has a very large overhead imposed by heavy zero-knowledge proofs. A proof-of-concept implementation is available in Sage and takes around one second for an offline computation, being around nine times faster than OPUS. However,

Table 3: Communication and computation complexity estimates of the protocols with different concrete parameters.

		protocol	rounds	comm. cost	isog. comp.	model (C-S)
$\gamma = 128$	$\sigma = 512$	NR-OT	2	21 kB	624	◐-◐
		NR-OT	4	51 kB	1410	●-◐
		OPUS	258	25 kB	386	◐-◐
$\gamma = 256$	$\sigma = 2048$	NR-OT	2	148 kB	1282	◐-◐
		NR-OT	4	369 kB	2818	●-◐
		OPUS	514	197 kB	770	◐-◐
$\gamma = 256$	$\sigma = 5280$	NR-OT	2	355 kB	1282	◐-◐
		NR-OT	4	886 kB	2818	●-◐
		OPUS	514	508 kB	770	◐-◐

the implementation is not necessarily complete, as it omits proofs and samples from a uniform distribution instead of a Gaussian distribution.

A newer lattice OPRF [ADDG23] significantly improves the communication cost in a malicious setting. The provided implementation in Rust does not include the non-interactive zero-knowledge proofs needed for a malicious client security and therefore is only semi-honest, while the communication estimates in Table 4 include proofs from a malicious client. Comparing the runtime of OPUS to [ADDG23] is a bit more nuanced. While the former needs $\approx$ 15s for the key generation, CSIDH is vastly faster, as it only requires 0.14ms for the same operation. The communication complexity of the lattice OPRF is also largely dominated by the key generation, which accounts for 108.5 MB of the communication cost. For the actual OPRF, only 36kB of communication are necessary, which is slightly more than OPUS. A big advantage of the construction is the lower round complexity.

Using preprocessing and dedicated symmetric primitives, [DGH$^+$21] produce a very efficient, semi-honest OPRF using preprocessing. They also require a trusted third party to generate correlated randomness. While benchmarks exist, the implementation is unfortunately not available to the public.

A different path is taken by [SHB23], who use their result that key-recovery of the Legendre PRF is equivalent to solving sparse multivariate equations over a prime field to construct an OPRF. It requires a preprocessing step to distribute correlated randomness amongst the participants of the protocol.

Table 4: Comparison with all other post-quantum OPRF proposals. DM denotes the dark matter PRF [BIP+18, CCKK21]. The instances aim at a security level of roughly 128 bits and use $\log_2 p = 512$ for the isogeny protocols.

work	assumption	rounds	comm. cost	model (C-S)	no preproc.	no trusted setup	full impl. available
[ADDS21]	R(LWE)+SIS	2	≈ 2MB	◐-◐	✓	✓	✓
[ADDS21]	R(LWE)+SIS	2	> 128GB	●-●	✓	✓	✗
[SHB23]	multivariate	3	$\gamma \cdot 13$ kB	◐-◐	✗	✓	✗
[DGH+21]	DM	2	308 B	◐-◐	✗	✗	✗
[ADDG23]	DM+lattices	2	≈ 108.5MB	●-◐	✓	✓	✓
[ADDG23]	DM+lattices	2	≈ 211.3MB	●-◐	✓	✓	✗
[Bas23]	Isogenies $\mathbb{F}_{p^2}$	2	3.0MB	●-●	✓	✗	✗
NR-OT	CSI-FiSh	2	20.54 kB	◐-◐	✓	✗	✗
NR-OT	CSI-FiSh	4	34.88 kB	●-◐	✓	✗	✗
OPUS	CSIDH	258	24.7 kB	◐-◐	✓	✓	✓

8 Conclusion

In this paper, we have shown that the computational complexity of Naor-Reingold OPRFs can be significantly reduced by using properties of the CSIDH group action. We introduced OPUS, an OPRF that gains its hardness directly from the underlying CSIDH group action. The new construction digs deeper into the generic construction of Naor-Reingold protocols, which traditionally use oblivious transfer to send blinded private keys. We found a generic way to directly use oblivious transfer-style computation to save several computations and remove the need for a trusted setup. In comparison to previous work, OPUS has three strong advantages: First, it can be used stand-alone without requiring any trusted setup. The only hardness assumption is CSIDH, which is fewer than in the previous generic proposal [BKW20] from CSIDH. Second, the simple structure also makes a compelling argument for the security analysis of OPUS. In addition, the extension to threshold and distributed OPRFs is straightforward. Third, OPUS requires 40% fewer isogeny computations than the best previous CSIDH-based OPRF proposals as shown in Table 3. This is compelling for constrained devices, e.g. IoT devices with reasonably good internet access.

When using no preprocessing, no trusted setup, and a semi-honest client and server, OPUS requires 83× less communication than the next-best approach which uses LWR. The main drawback of our construction is the large number of rounds, which can be amortized over several executions.

We also revisited the only other previous proposal from CSIDH that survived cryptanalysis so far [BKW20], and showed that the implementation is more complex than described in the original paper: A straight-forward implementation leaks the entire server key after a few evaluations. To secure the construction, it is necessary to use CSI-FiSh, which introduces several new hardness assumptions, concretely lattice assumptions for either rejection sampling or reducing

the private key. Besides more hardness assumptions, this also adds additional overhead compared to the previous proposal.

Of independent interest, we also discuss the Naor-Reingold PRF in CSIDH further and give a concrete strategy that gives rise to optimizations in all of our protocols and also enables somewhat fast offline computation of both our novel OPRF and the Naor-Reingold OPRF. All the measurements and the code to obtain them are available at `https://github.com/meyira/OIDA`.

Future Work While our results are immediately useful for a variety of protocols requiring OPRFs, the slow group action is still hindering large-scale deployment. Based on our findings, we envision future studies for the applicability of OPUS to more advanced protocols, especially in settings with low bandwidth. We anticipate that there may be some use having a verifiable delay both due to the sequential computation and the physical limitations of the network.

We already discussed more nuanced possible extensions, amongst them POPRFs, threshold OPRFs and updatable OPRFs in Section 4.2. In addition, the current implementation of OPUS is neither optimized nor side-channel free. Concretely, OPUS only requires side-channel free key addition and group actions, as well as the conditional assignment of E_{client}, to become side-channel free. This is especially true for a group action with a larger security parameter. On a theoretical side, elliptic curves with trusted setup over $\mathbb{F}_p$ would greatly add to the current research, as it eases concretizing the overhead of the OT for the NR-OT proposal over CSIDH and would enable a variety of simpler protocols with fewer rounds.

Acknowledgements We wholeheartedly thank Carsten Baum for many helpful discussions concerning OPUS in particular and OPRFs in general. In addition, we are gracious of the very helpful feedback of the reviewers of PKC2022 on an earlier draft of this work. Furthermore, we thank Serge Bazanski for some helpful suggestions that led to the creation of Figure 9. Finally, we thank the authors of [BKW20] for confirming our suspicions about the protocol and clarifying their instantiation.

References

[ADDG23] Martin R. Albrecht, Alex Davidson, Amit Deo, and Daniel Gardham. Crypto dark matter on the torus: Oblivious PRFs from shallow PRFs and FHE. Cryptology ePrint Archive, Report 2023/232, 2023. `https://eprint.iacr.org/2023/232`.

[ADDS21] Martin R. Albrecht, Alex Davidson, Amit Deo, and Nigel P. Smart. Round-optimal verifiable oblivious pseudorandom functions from ideal lattices. In Juan Garay, editor, PKC 2021, Part II, volume 12711 of LNCS, pages 261–289. Springer, Heidelberg, May 2021.

[ADMP20] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In Shiho Moriai and Huaxiong Wang, editors, ASIACRYPT 2020, Part II, volume 12492 of LNCS, pages 411–439. Springer, Heidelberg, December 2020.

[Bas23] Andrea Basso. A post-quantum round-optimal oblivious PRF from isogenies. Cryptology ePrint Archive, Report 2023/225, 2023. https://eprint.iacr.org/2023/225.

[BCC+23] Andrea Basso, Giulio Codogni, Deirdre Connolly, Luca De Feo, Tako Boris Fouotsa, Guido Maria Lido, Travis Morrison, Lorenz Panny, Sikhar Patranabis, and Benjamin Wesolowski. Supersingular curves you can trust. In Carmit Hazay and Martijn Stam, editors, Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part II, volume 14005 of Lecture Notes in Computer Science, pages 405–437. Springer, 2023.

[BIP+18] Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring crypto dark matter: New simple PRF candidates and their applications. In Amos Beimel and Stefan Dziembowski, editors, TCC 2018, Part II, volume 11240 of LNCS, pages 699–729. Springer, Heidelberg, November 2018.

[BKM+21] Andrea Basso, Péter Kutas, Simon-Philipp Merz, Christophe Petit, and Antonio Sanso. Cryptanalysis of an oblivious PRF from supersingular isogenies. In Mehdi Tibouchi and Huaxiong Wang, editors, ASIACRYPT 2021, Part I, volume 13090 of LNCS, pages 160–184. Springer, Heidelberg, December 2021.

[BKV19] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In Steven D. Galbraith and Shiho Moriai, editors, ASIACRYPT 2019, Part I, volume 11921 of LNCS, pages 227–247. Springer, Heidelberg, December 2019.

[BKW20] Dan Boneh, Dmitry Kogan, and Katharine Woo. Oblivious pseudorandom functions from isogenies. In Shiho Moriai and Huaxiong Wang, editors, ASIACRYPT 2020, Part II, volume 12492 of LNCS, pages 520–550. Springer, Heidelberg, December 2020.

[BLMP19] Daniel J. Bernstein, Tanja Lange, Chloe Martindale, and Lorenz Panny. Quantum circuits for the CSIDH: Optimizing quantum evaluation of isogenies. In Yuval Ishai and Vincent Rijmen, editors, EUROCRYPT 2019, Part II, volume 11477 of LNCS, pages 409–441. Springer, Heidelberg, May 2019.

[BS20] Xavier Bonnetain and André Schrottenloher. Quantum security analysis of CSIDH. In Anne Canteaut and Yuval Ishai, editors, EUROCRYPT 2020, Part II, volume 12106 of LNCS, pages 493–522. Springer, Heidelberg, May 2020.

[CCKK21] Jung Hee Cheon, Wonhee Cho, Jeong Han Kim, and Jiseung Kim. Adventures in crypto dark matter: Attacks and fixes for weak pseudorandom functions. In Juan Garay, editor, PKC 2021, Part II, volume 12711 of LNCS, pages 739–760. Springer, Heidelberg, May 2021.

[CHL22] Sílvia Casacuberta, Julia Hesse, and Anja Lehmann. Sok: Oblivious pseudorandom functions. In 7th IEEE European Symposium on Security and Privacy, EuroS&P 2022, Genoa, Italy, June 6-10, 2022, pages 625–646. IEEE, 2022.

[CLM+18] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In Thomas Peyrin and Steven Galbraith, editors, ASIACRYPT 2018, Part III, volume 11274 of LNCS, pages 395–427. Springer, Heidelberg, December 2018.

[Cou06] Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291, 2006. `https://eprint.iacr.org/2006/291`.

[CSCJR22] Jorge Chávez-Saab, Jesús-Javier Chi-Domínguez, Samuel Jaques, and Francisco Rodríguez-Henríquez. The SQALE of CSIDH: sublinear Vélu quantum-resistant isogeny action with low exponents. Journal of Cryptographic Engineering, 12(3):349–368, September 2022.

[DFHSW22] Alex Davidson, Armando Faz-Hernández, Nick Sullivan, and Christopher A. Wood. Oblivious Pseudorandom Functions (OPRFs) using Prime-Order Groups. Internet-Draft draft-irtf-cfrg-voprf-12, Internet Engineering Task Force, August 2022. Work in Progress.

[DG19] Luca De Feo and Steven D. Galbraith. SeaSign: Compact isogeny signatures from class group actions. In Yuval Ishai and Vincent Rijmen, editors, EUROCRYPT 2019, Part III, volume 11478 of LNCS, pages 759–789. Springer, Heidelberg, May 2019.

[DGH+21] Itai Dinur, Steven Goldfeder, Tzipora Halevi, Yuval Ishai, Mahimna Kelkar, Vivek Sharma, and Greg Zaverucha. MPC-friendly symmetric cryptography from alternating moduli: Candidates, protocols, and applications. In Tal Malkin and Chris Peikert, editors, CRYPTO 2021, Part IV, volume 12828 of LNCS, pages 517–547, Virtual Event, August 2021. Springer, Heidelberg.

[DGS+18] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. PoPETs, 2018(3):164–180, July 2018.

[DM20] Luca De Feo and Michael Meyer. Threshold schemes from isogeny assumptions. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, PKC 2020, Part II, volume 12111 of LNCS, pages 187–212. Springer, Heidelberg, May 2020.

[dSGOPS20] Cyprien Delpech de Saint Guilhem, Emmanuela Orsini, Christophe Petit, and Nigel P. Smart. Semi-commutative masking: A framework for isogeny-based protocols, with an application to fully secure two-round isogeny-based OT. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, Cryptology and Network Security - 19th International Conference, CANS 2020, Vienna, Austria, December 14-16, 2020, Proceedings, volume 12579 of Lecture Notes in Computer Science, pages 235–258. Springer, 2020.

[FIPR05] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In Joe Kilian, editor, TCC 2005, volume 3378 of LNCS, pages 303–324. Springer, Heidelberg, February 2005.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In G. R. Blakley and David Chaum, editors, CRYPTO'84, volume 196 of LNCS, pages 276–288. Springer, Heidelberg, August 1984.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. Journal of the ACM, 33(4):792–807, October 1986.

[Hun] Troy Hunt. Pwned websites. see `https://haveibeenpwned.com/pwnedwebsites`.

[KRS+19] Daniel Kales, Christian Rechberger, Thomas Schneider, Matthias Senker, and Christian Weinert. Mobile private contact discovery at scale. In Nadia Heninger and Patrick Traynor, editors, USENIX Security 2019, pages 1447–1464. USENIX Association, August 2019.

[LGd21] Yi-Fu Lai, Steven D. Galbraith, and Cyprien de Saint Guilhem. Compact, efficient and UC-secure isogeny-based oblivious transfer. In Anne Canteaut and François-Xavier Standaert, editors, EUROCRYPT 2021, Part I, volume 12696 of LNCS, pages 213–241. Springer, Heidelberg, October 2021.

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, ASIACRYPT 2009, volume 5912 of LNCS, pages 598–616. Springer, Heidelberg, December 2009.

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. Journal of the ACM, 51(2):231–262, 2004.

[Pei20] Chris Peikert. He gives C-sieves on the CSIDH. In Anne Canteaut and Yuval Ishai, editors, EUROCRYPT 2020, Part II, volume 12106 of LNCS, pages 463–492. Springer, Heidelberg, May 2020.

[RS06] Alexander Rostovtsev and Anton Stolbunov. Public-Key Cryptosystem Based On Isogenies. Cryptology ePrint Archive, Report 2006/145, 2006. `https://eprint.iacr.org/2006/145`.

[Sha79] Adi Shamir. How to share a secret. Communications of the Association for Computing Machinery, 22(11):612–613, November 1979.

[SHB23] István András Seres, Máté Horváth, and Péter Burcs. The legendre pseudorandom function as a multivariate quadratic cryptosystem: security and applications. In AAECC. Springer, 01 2023.

[Sil86] Joseph H. Silverman. The arithmetic of elliptic curves, volume 106 of Graduate texts in mathematics. Springer, 1986.

[Sut12] Andrew V. Sutherland. Identifying supersingular elliptic curves. LMS Journal of Computation and Mathematics, 15:317–325, 2012.

[Vél71] J. Vélu. Isogénies entre courbes elliptiques. Comptes-Rendus de l'Académie des Sciences, Série I, 273:238–241, juillet 1971.

A Artifact Description

We provide all benchmark-generating files as well as example instantiations of OPUS in the artifact and plan to publicly host the code with the publishing of this paper. A Makefile is provided in the artifact for easy compilation and linking. The code is in the subdirectory `code/` and the corresponding CSV files, if needed, are in `csv-files/`

A.1 Generation of Figure 4

The data used in Figure 4 is in `noopt.csv`, and the code is in `prf.c`.

A.2 Generation of Figure 5

The data for Figure 5 is in `updatable.csv`, and the code to generate it is in `updatable.c`. A verification routine is commented out as it is a bit annoying during benchmarking.

A.3 Generation of Table 1

`opus.c` performs local computations of OPUS. The code was used to generate Table 1.

A.4 Generation of Figure 10

The online evaluation of OPUS with different servers in London in Figure 9 was generated with averages over a few runs. The data can be seen in `online.csv` in the artifact. The client used single-threaded sockets for evaluation, as visible in `client.c`. The server is multithreaded and can be seen in `server.c`. Due to the high network latency and multithreading, no effect on concurrent execution was visible, but for clear benchmarks, we refrained from concurrent execution to have a comparable result.

The server has a KAT functionality built-in but commented out, where for a fixed message both the client and the server should obtain the same result. This sanity check may be of use to implementors.

We also benchmarked Sao Paulo, where the online evaluation and ping were (34.54s, 190ms), respectively, Hongkong with (43.43s, 257ms) and Frankfurt with (11.91s, 12.6ms). They were omitted as they made the graph unreadable, being too close to Santiago de Chile, Sydney, and the Netherlands, respectively.

A.5 Approximation for Key Leakage

`leak_OPRF_key_csidh.py` gives a rough approximation on how long it would take for a simple attack on the NR-OT OPRF as described in Section 3.3.