OPRFs from Isogenies: Designs and Analysis

Lena Heimberger[1], Tobias Hennerbichler[1], Fredrik Meisingseth[1,2], Sebastian Ramacher[3], Christian Rechberger[1]

[1]Graz University of Technology, [2] Know-Center, [3] AIT Austrian Institute of Technology
lena.heimberger@tugraz.at

September 28, 2023

Abstract

Oblivious Pseudorandom Functions (OPRFs) 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, and of those even fewer are practical for modern-day applications. In this work, we focus on isogeny group actions, as the associated low bandwidth leads to efficient constructions. Our results focus on the Naor-Reingold OPRF. We introduce OPUS, a novel OPRF from isogenies without oblivious transfer, and show efficient evaluations of the Naor-Reingold PRF using CSIDH and CSI-FiSh. Additionally, we analyze a previous proposal of a CSIDH-based OPRF and that the straightforward instantiation of the protocol leaks the server's private key. As a result, we propose mitigations to address those shortcomings, which require additional hardness assumptions. Our results report a very competitive protocol when combined with lattices for Oblivious Transfer.

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, and the repaired construction. Finally, we demonstrate the efficiency of OPUS and the generic NR-OT in two use cases: first, we instantiate OPAQUE, a protocol for asymmetric authenticated key exchange. Compared to classical elliptic curve cryptography, this results in less than 100 × longer computation on average and around 1000× more communication overhead. Second, we perform an unbalanced private set intersection and show that the communication overhead can be roughly the same when using isogenies or elliptic curves, at the cost of much higher runtime. Conversely, for sets of the size 2^{10}, we report a runtime around 200× slower than the elliptic curve PSI. This concretizes

the overhead of performing PSI and using OPAQUE with isogenies for the first time.

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. The OPRF is secure when the client learns nothing about the key, and the server learns nothing about the output or the client input.

This basic functionality gives rise to various applications. Consider password authentication: To prove the knowledge of a pre-registered password, the client transmits their password, ideally in a salted and hashed form. The server checks the transmitted password against a stored record and authenticates the client if the record matches the password. However, passwords notoriously lack entropy and may be recovered from a server record in the event of a breach. In addition, this ideal setting is not always the case, as attacks leaking cleartext passwords are still common. For example, the PwnedPasswords web page [Hun] consolidates breaches of passwords and finds over 90 matches when searching for *plain text* breaches.

This attack vector can be mitigated by never storing passwords on a server in the first place. A great example of a protocol solving the password storage problem is OPAQUE, an asymmetric password-authenticated key agreement protocol for which standardization efforts are ongoing at the CFRG [DFHSW22].

Using OPRFs to add privacy to applications expands beyond passwords. A nice use case is private set intersection (PSI), where two parties with respective datasets wish to compute the overlapping elements in both sets without revealing their non-shared elements. This can be used for private contact discovery [KRS+19] to protect 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 OPRFs from classical primitives, efficient and secure OPRFs from post-quantum hardness assumptions remain an open question. A nice primitive 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 uniform sampling for the keys as used in the signature scheme CSI-FiSh [BKV19]. We combine the OPRF with a lattice-based Oblivious Transfer protocol to achieve a relatively fast construction that computes the OPRF in under 100 ms online time. Of independent interest, we report that the Naor-Reingold PRF is nearly constant-time with respect to the input length when using the lattice reductions of CSI-FiSh.

Based on the work on this OPRF, we introduce OPUS, a novel construction that

only uses CSIDH operations. It efficiently computes the Naor-Reingold OPRF while only using 60% of the group actions of the previous proposal, without needing a trusted setup.

Using the two OPRFs, we present the first post-quantum implementation of OPAQUE using two isogeny-based OPRFs. In addition, we performed private set intersection with both OPRFs.

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 $*$, which iteratively computes the ℓ_i isogenies. It acts 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, verifying a curve is supersingular simply

requires 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 2.3 onwards, as it gives rise to 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 Assumptions

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.*

We recall the computational CSIDH problem from [CLM+18].

Problem 2 (Computational CSIDH Problem)**.** *Given curves $E \in \mathcal{E}$, $\mathsf{r} * E \in \mathcal{E}$, and $\mathsf{s} * E \in \mathcal{E}$ where $\mathsf{r}, \mathsf{s} \in Cl(\mathcal{O})$, find $E' \in \mathcal{E}$ such that $E' = \mathsf{r} * \mathsf{s} * E$.*

Finally, we recall the decisional CSIDH problem from [EKP20]:

Problem 3. *Decisional CSIDH Problem Given the set of curves $\mathcal{E}$ and the ideal class group $Cl(\mathcal{O})$, the decisional CSIDH (D-CSIDH) problem asks to distinguish between the following two distributions:*

- *$(E, H, a * E, a * H)$ with $E, H \leftarrow\!\!\$\ \mathcal{E}$ and $a \leftarrow\!\!\$\ Cl(\mathcal{O})$.*
- *(E, H, E', H') where $E, H, E', H' \leftarrow\!\!\$\ \mathcal{E}$.*

If for all PPT adversaries $\mathcal{A}$, the advantage in distinguishing the two distributions is negligible, we say that the C-CSIDH assumption holds.

2.1.5 Parameterization and Security

The size of the prime p denotes the security parameter of CSIDH. There is heavy disagreement in the literature on the secure parameterization of CSIDH [BLMP19, BS20, Pei20], as 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.

2.2 CSI-FiSh

Building on CSIDH, the signature scheme CSI-FiSh introduces a uniform representation of the class group elements. In their paper, this is necessary for the Fiat-Shamir transformation to obtain a signature scheme, 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 and reduces the key after the arithmetic operation to avoid leakage. Due to the different distribution of the class group ideals, the group action is around 15% slower.

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

The Naor-Reingold PRF [NR04] is a generic construction for PRFs from Abelian group actions that 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.

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

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.*

2.4 Oblivious Transfer and Naor-Reingold OPRF

The NR-PRF gives rise to oblivious evaluation using oblivious transfer (OT). 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 and is secure when the client learns nothing about m_{1-c} and the server learns nothing about c.

To compute the NR-PRF obliviously using OT, the input X is bit-decomposed into $X = [x_1, \ldots, x_n]$ to use as an input for the OT. The server samples n blinding elements $[r_1, \ldots, r_n]$ and inputs $r_i, k_i \circ r_i$ to the OT, with r_i perfectly hiding k_i. The client queries the OT with each x_i to obtain $k_i{}_i^x \circ r_i$ and aggregates all results with the group action to obtain the blinded group element ${k_1}^{x_1} \circ r_1 \circ \ldots \circ {k_n}^{x_n} \circ r_n$. To finalize the computation, the server evaluates the inverse of all blinding elements with the key and sends the result $k_0 \circ r_1^{-1} \circ \ldots \circ r_n^{-1}$ to the client, who performs a final group action with the finalization element and the blinded group element to obtain the result of the NR-PRF.

2.5 Notation

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 $\mathbf{a} - \mathbf{b}$.

We denote the sequential application of the group action $\mathsf{csidh}(\mathsf{csidh}(E, \mathbf{a}), \mathbf{b})$ as $\mathbf{b} * (\mathbf{a} * E)$. Due to the commutativity of CSIDH, this is also equivalent to $(\mathbf{a} + \mathbf{b}) * E$. 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. 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 define a *round* as one message from a party to another, i.e. a two-party protocol with two rounds of communication is round-optimal. 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.

2.6 Benchmarks

All benchmarks, unless specified otherwise, are averaged over 100 executions with random input and have been run on a computer with an AMD Ryzen 7 PRO 4750U Processor with a fixed processor speed at 1.7 GHz and 24 GiB RAM, under the Linux kernel 6.1.44-1-lts. We will refer to this setup as the `test machine`. Unless otherwise stated, the input length to the OPRF is 128 bits.

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

Previous work [BKW20] describes the Naor-Reingold OPRF for 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 of both the NR-PRF and the NR-OPRF from CSIDH. More concretely, we show the naïve instantiation of the OPRF in Section 3.1 leads to a full key leak in a passive attack in Section 3.2.3 and propose a mitigation in Section 3.3.

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

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

Figure 2: Naor-Reingold PRF from CSIDH using E_0 as a starting curve. We use $k_i^{x_i}$ as a shorthand notation for *perform the group action with* k_i *if and only if* x_i *is set*

of the PRF outlined in Figure 2 is prohibitively slow, requiring $n+1$ sequential group actions to compute the PRF for n input bits. A recent paper [ADMP20] describes an effective way to evaluate the PRF by splitting the evaluation into

two parts: 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 using the aggregated key elements in a second step on the base 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 reduces the evaluation time. This is due to two factors: one, the key coefficients are in the range

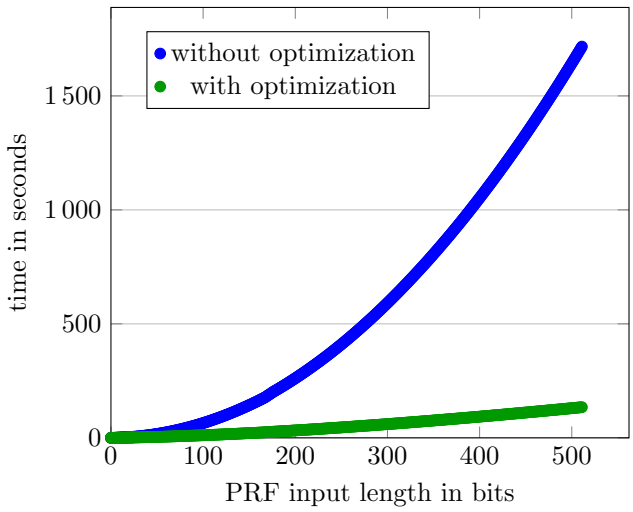


Figure 4: Runtime divergence between the traditional Naor-Reingold CSIDH PRF in blue and the same PRF with our optimization in green for different bit lengths.

[1] All CSIDH benchmarks use the reference implementation from `https://yx7.cc/code/csidh/csidh-latest.tar.xz`, which is from 27-06-2021.

$[-5, 5]$ and will partially cancel out when added, reducing the required steps on the isogeny graph. Two, the optimization 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

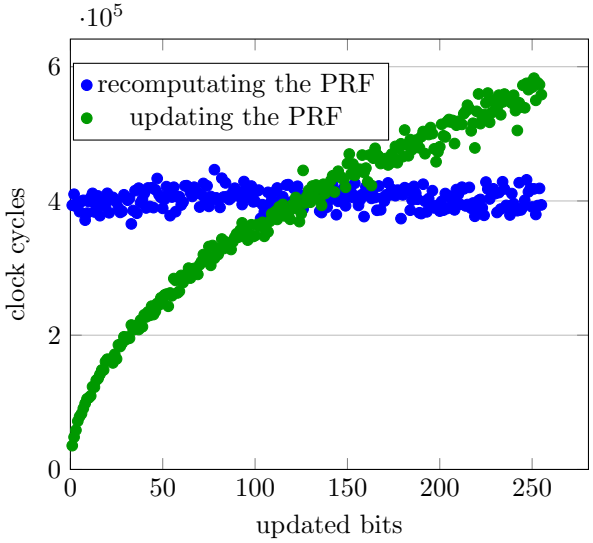


Figure 5: Runtime divergence between updating x bits of the PRF vs. recomputing the full 256 bits of the PRF.

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 in the runtime is due to the non-uniform keys in CSIDH.

3.1.1 Instantiation from CSI-FiSh

The PRF is even more efficient with CSI-FiSh, as the keys can be added and then reduced modulo the class group number:

The reduction step leads to an almost constant-time computation. In Figure 7, we show the improvement in runtime when using a reduction, leading to an almost constant time complexity when computing the PRF, independent of

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

Figure 6: 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 input. More concretely, the difference between the lowest and the highest execution time is 0.0032s for the optimized variant and 0.4377s for the aggregation variant.

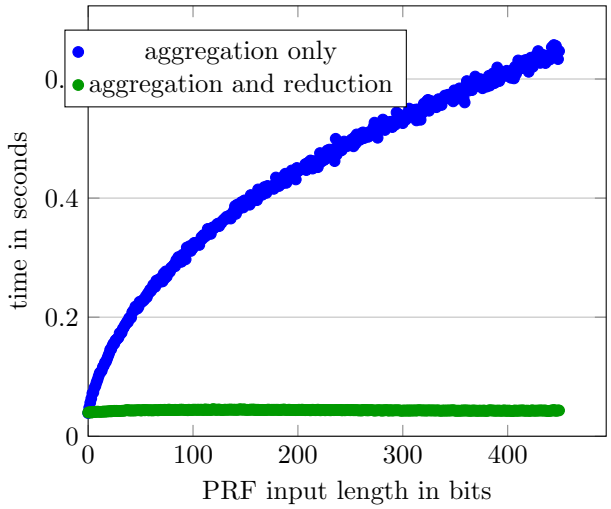


Figure 7: Comparing PRF runtimes using aggregation only and aggregation and a reduction modulo the class group number before applying the group action.

3.2 Oblivious NR-PRF from CSIDH

The OPRF in [BKW20] is not rigorously described; they initially give a description of the NR-PRF in Protocol 24 of the same paper. In a later paragraph, they state instantiating said protocol with CSIDH results in a NR-OPRF similar to the protocol in Section 2.3. Since the protocol uses OT, we will call it NR-OT henceforth. Using our addition trick, a correct intuition to compute the OPRF is to instantiate the OT with $(\mathbf{r_i}, \mathbf{k_i} + \mathbf{r_i})$ and finalizing the OT by sending $\mathbf{k_0} * \sum_{i=1}^{n} -\mathbf{r_i}$. We sketch this protocol in Figure 15.

3.2.1 Analyzing the Construction

While the OPRF above produces a correct result, due to the non-uniform representation of the CSIDH private key, the construction leaks the server key. [2] A passive adversary, that is, an adversary who carries out the protocol faithfully, can observe the distribution of the blinded keys.

3.2.2 Key Leakage Example

Consider blinding a key coefficient $k_i = 5$ with a random element r_i. Regardless of the value of r_i, the blinded element $r_i + k_i$ is always within the range $[0, 10]$, as $r_i \in [-5, 5]$. Over several iterations, r_i will change and reveal more and more information about the key, giving the information outright when the blinded result is 10 or -10, since this is only possible when both the key and the random element are 5 or -5, respectively.

3.2.3 Full Attack

The passive adversary with the goal to leak the private key $\mathbf{k}$ monitors the the blinded results and observes that the static key becomes immediately clear when the highest and lowest result of a key coefficient have a difference of $2 *$ MAX_COEFFICIENT $= 10$, as k_i is the highest coefficient minus MAX_COEFFICIENT.

For the attack, the adversary records the lowest and highest results of a blinded key coefficient seen so far, and stops once the difference is high enough to obtain k_i. Note that this strategy does not take into account more sophisticated methods such as guessing when only a few bits of a key are missing.

To simulate such an attacker, we generated a static key $\mathbf{k}$ and an oracle that returns a blinded $\mathbf{k}$ when queried. Using the oracle, the adversary checks if -10 or 10 occurred or the interval between the highest and the lowest query is large enough to determine the key coefficient. On average, the entire key $\mathbf{k}$ is leaked after 65 evaluations. The example implementation is available with this paper.

3.3 Fixing the NR-OPRF

The non-uniform key distribution has already been discussed in the context signatures schemes. The signature scheme SeaSign [DG19] mitigates the non-uniform mitigation by rejection sampling, concretely using the Fiat-Shamir transformation with aborts [Lyu09]. To translate the technique to the CSIDH setting, SeaSign uses somewhat short, long-term secret keys $\mathbf{k}$ with coefficients $k_i \in [-B, B]^k$ for some B and large, ephemeral secret keys $\mathbf{r}$ with each coefficient $r_i \in [-(\delta+1)B, (\delta+1)B]^k$, rejecting any $\mathbf{r}$ where the vector $\mathbf{r} - \mathbf{k}$ contains a coefficient is outside of the range $[-\delta B, \delta B]$. In the NR-OT setting, the long-term sender keys are the short keys $\mathbf{s}$ and the ephemeral keys are sampled as $\mathbf{r}$.

[2] In personal communication, authors of [BKW20] confirmed that the 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".

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

Most of these issues are mitigated by using the sampling algorithm from the signature scheme CSI-FiSh [BKV19] introduced in Section 2.2. The protocol in Figure 15 would largely remain the same, with $\mathbf{k_i}+\mathbf{r_i}$ being a reduced element of the class group.

Another roadblock on the way to a secure NR-OT instantiation is the underlying OT. The estimations for the communication complexity of the NR-OT [BKW20] use an isogeny-based OT protocol [LGD21] that requires a supersingular curve with an unknown endomorphism ring, also called SECUER. While a recent paper [BCC+23] proposes an algorithm for the generation for supersingular curves over $\mathbb{F}_{p^2}$, there are no known efficient algorithms for the curves over $\mathbb{F}_p$ used by CSIDH, which is denoted as an open problem in the same paper. Therefore, using the OPRF protocol requires either an efficient construction of SECUERs over $\mathbb{F}_p$ or a different OT protocol without a trusted setup.

Alternate OT protocols using CSIDH other challenges: The semi-honest protocol of [dSGOPS20] gives similar performance to the OT protocol of [LGD21], but requiring two trusted curves for the setup. A good alternative may be the single-bit OT of [ADMP20], which requires a key distribution closer to uniform than CSIDH and therefore uses the CSI-FiSh key sampling algorithm for the entire protocol. The main issue with this protocol is that the number of isogeny computations depends on the length of the client input and the bitlength of the input $\log_2 p = \sigma$. The overall number of isogeny computations would be $\gamma(5\sigma + 5)$. For an input length of 128 bits and a key size of 256 bits, this would amount to 164480 isogeny computations, which is prohibitive.

To instantiate the protocol regardless, an OT without CSIDH seems necessary. We chose a two-round protocol based on additive homomorphic encryption [BDK+20] to model the OT, as it provides an implementation and is round-optimal. In addition, the protocol offers batching, making it more efficient for multiple OT invocations, and expects the input to be given as a GMP integer, which is how CSI-FiSh encodes the private key. The protocol is implemented in C++ using Microsoft SEAL [SEA21] for the homomorphic operations. Using the BFV [Bra12] [FV12] scheme, it follows three steps, with homomorphic operations on encrypted messages denoted in $\square$.

1. The client encrypts their choice bit $c_b = \mathrm{Enc}(pk, b)$ and sends it to the server.

2. The server computes $c_{m_b} = (m_0 \boxdot (1 \boxminus c_b)) \boxplus (m_1 \boxdot c_b)$ and sends c_{m_b} to the client.

3. The client decrypts the ciphertext to obtain $m_b = \mathrm{Dec}(sk, c_{m_b})$

Using the OT and CSI-FiSh, the full protocol is displayed in Figure 8.

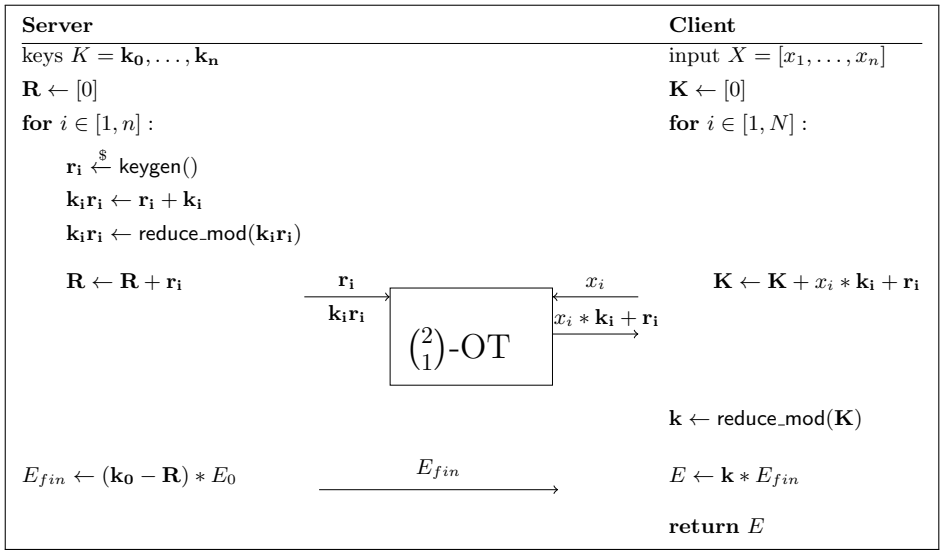


Figure 8: Full protocol of evaluating the NR-OPRF with CSI-FiSh and N OT calls. The function reduce_mod describes the reduction modulo the class group number.

3.3.1 Performance

Using the lattice-based OT, the NR-OT OPRF becomes relatively efficient. This is due to two factors: first, the added keys are reduced modulo the class number, which results in a very fast PRF runtime, see Section 3.1.1. This results in a protocol that only requires two group actions to complete. Second, while the lattice OT requires a lot of communication, it is relatively fast.

Table 1: Comparison between PRF and OPRF execution time locally on the test machine for our NR-OT OPRF. The network traffic is always denoted as sent kilobytes. OT keygen is a separate column for key generation measuring the client communication and computation time.

Input-length	Keygen PRF	Comp. PRF	Client	Server	OT keygen
128	196ms	43ms	91ms 128 kiB	95ms 256 kiB	429ms 256 kiB
256	377ms	43ms	97ms 256 kiB	102ms 512 kiB	428ms 256 kiB
512	779ms	45ms	107ms 384 kiB	110ms 768 kiB	427ms 256 kiB

3.3.2 Conclusion

The construction repairs the issues from the initial proposal [BKW20], namely by using an OT protocol that does not require a trusted setup and using the sampling approach from CSI-FiSh for uniform keys. This introduces two new issues: First, the OT protocol allows the client's choice bit to be neither 0 nor 1, which may result in a response that is a superposition of messages. Hence, the security model is weaker, as a semi-honest client would only be passively secure. Second, when using uniform sampling, the class group structure is only available for primes of length 512 [BKV19] or 1024 [DFK+23], which may not provide a sufficient security margin as discussed in Section 2.1.5.

4 OPUS: Oblivious Pseudorandom Function using CSIDH

While the above construction is relatively efficient, it would be nice to build a similar OPRF exclusively from isogenies without the need for any lattice reductions. To avoid sending any private keys over the network, we propose OPUS, a novel OPRF 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, see Figure 9.

4.1 Efficiency

Once again, the OPRF is made more efficient with the addition trick from Section 2.3, as both client and server aggregate the blinding keys in vector R to quickly reduce the number of group actions. Overall, OPUS needs $2n+1$ group action computations for the server and $n+1$ for the client. Experimental runtimes can be found in Table 2, and concrete runtimes in Table 8. 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.

Aside from the isogeny computations, the main performance issue in OPUS is the large number of rounds. To address this concern, we rented virtual machines around the world and used them as clients performing OPUS with a server in London. As clear from Figure 10, the runtime of OPUS directly corresponds

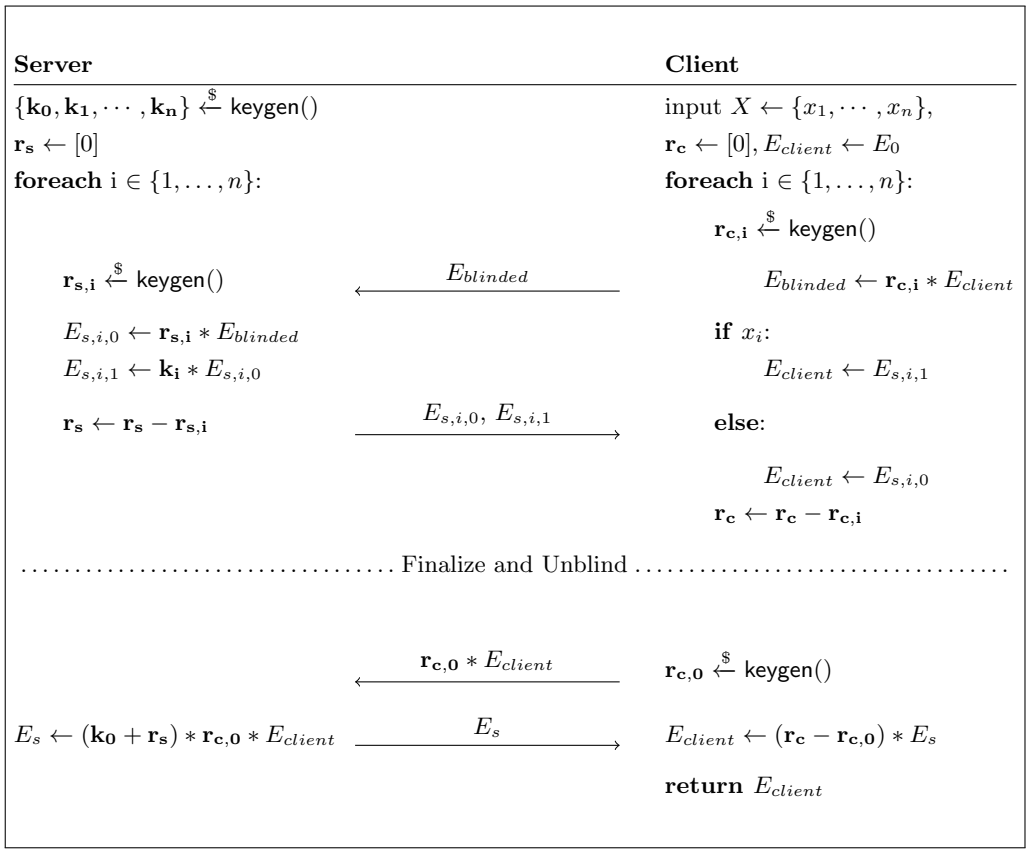


Figure 9: The full protocol of our novel OPRF *OPUS*.

Table 2: Comparison of OPUS complexity on the test machine. The overall time is the addition of the time from the client and the server, as the protocol is sequential.

Bit-length	Keygen PRF	Comp. PRF	Client	Server	Overall
128	0.14ms	172ms	3.16s 64.5 kiB	6.02s 128.5 kiB	9.17s 193 kiB
256	0.28ms	248ms	6.27s 128.5 kiB	12.13s 256.5 kiB	18.40s 385 kiB
512	0.60ms	353ms	12.37s 256.5 kiB	24.20s 513 kiB	36.58s 769 kiB

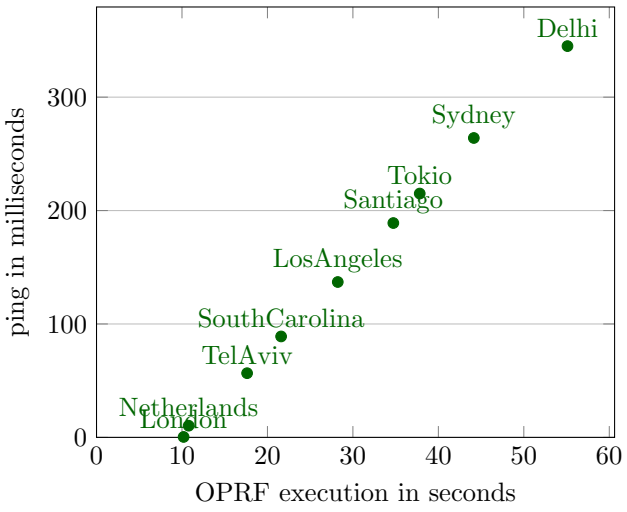


Figure 10: Online runtimes of different clients computing OPUS with a bit length of 128 with a server in London. All machines run on Debian 11 using the simplest Google Cloud instance.

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 Verifiability

Some protocols using OPRFs require the underlying OPRF to be *verifiable*, which means the functionality ensures a server uses a certain key that it previously committed to for the evaluation. This is necessary to ensure the same key was used for all clients, which avoids tagging individual users or sessions, as different keys may reveal the user's identity at a later stage of the protocol. Adding verifiability to OPUS is difficult as the communication is entirely over randomized curves, similar to the challenges imposed by the requirements for malicious security. Another OPRF based on isogenies over $\mathbb{F}_{p^2}$ [Bas23] uses a *proof of parallel isogeny*, which provides a zero-knowledge proof to show that two curves were computed by applying the same secret key to two starting curves and torsion points. Unfortunately, this does not carry over to CSIDH's $\mathbb{F}_p$ and cannot be applied OPUS or the NR-OT. A recent survey [BFGP23] details strategies and gives an overview of zero-knowledge proofs for isogenies. While it seems possible, we leave the task of constructing a verifiable OPRF for future work.

5 Security Analysis

To prove our novel OPRF secure against a semi-honest adversary in the ROM, we will first show that the OPUS is a PRF. For the security proof that OPUS is a secure OPRF, we refer the reader to Appendix D.

We now show that the protocol OPUS in Figure 9 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 the CSIDH-based* 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.*

Correctness of OPUS. Given input $X = (x_1, \ldots, x_n)$ and keys $K = (\mathbf{k_0}, \ldots, \mathbf{k_n})$, the client $\mathcal{C}$ initializes $E \leftarrow E_0$. For each $i \in [1, n]$, $\mathcal{C}$ generates a random key $\mathbf{r_{c,i}}$ and sends a randomized curve $\mathbf{r_{c,i}} * E$ to the server $\mathcal{S}$, which samples their randomness $\mathbf{r_{s,i}}$ and returns $E_{i,0} \leftarrow \mathbf{r_{s,i}} * E$ and $E_{i,1} \leftarrow \mathbf{k_i} * \mathbf{r_{s,i}} * E$ to $\mathcal{C}$. If $x_i = 1$, $\mathcal{C}$ sets $E \leftarrow E_{i,0}$ and $E \leftarrow E_{i,1}$ 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

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

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. □

Consequently, we obtain the following corollary from [BKW20, Theorem 23]:

Corollary 1. *Assuming computational CSIDH (cf. Problem 2) holds, then OPUS is a secure pseudorandom function.*

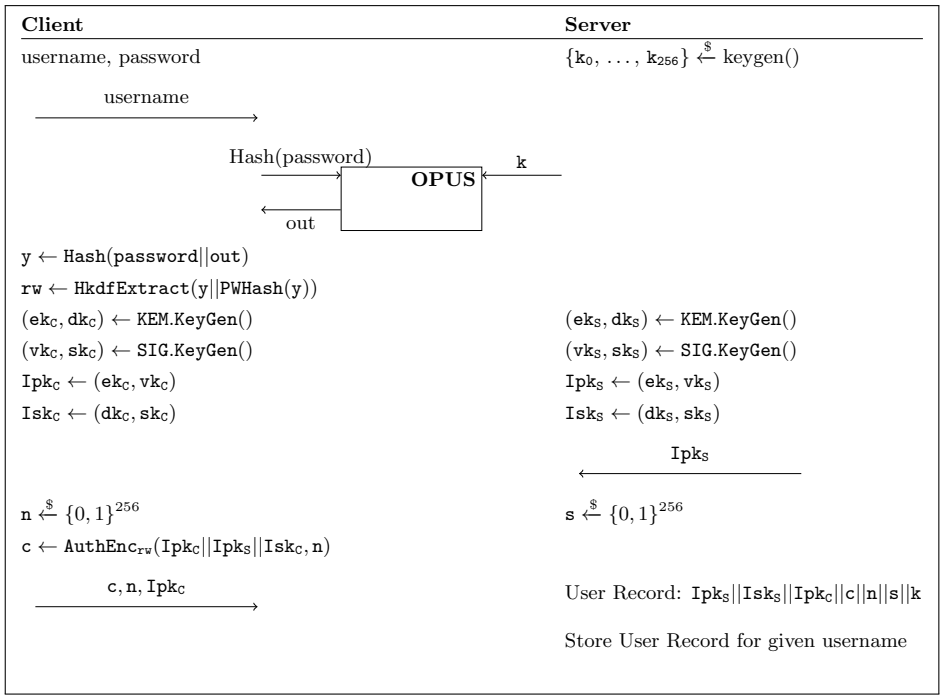
6 Case Study: OPAQUE

The OPAQUE [JKX18] protocol introduces a Password-Authenticated Key Exchange (PAKE) protocol that does not reveal the user's password to the server. Instead, it performs an OPRF calculation with the server, using the hash of the password as the user's input and a PRF key provided by the server. As previously mentioned, hiding the password from the server removes the threat of recovering the password in the case of a server compromise, by preventing pre-computation attacks.
OPAQUE consists of two phases: Password Registration and Password Authentication with Key Generation. Authentication and key generation are accomplished by either combining the OPRF with an asymmetric PAKE (aPAKE) or an Authenticated Key Exchange (AKE) protocol. In our implementation, we focus on the composition using the AKE protocol, since no CSIDH-based aPAKE protocols are available. During registration, both parties generate a long-term asymmetric keypair, later used during authentication to perform the AKE protocol. Using the output of the OPRF, the client derives a symmetric key and uses it to encrypt its private key. For simplicity, our implementation includes the client and server public key in the encryption process. The ciphertext is sent and stored on the server. During authentication the server fetches the ciphertext and sends it to the client, where it is decrypted after performing the OPRF again, requiring the user to only remember their password, but not the long-term keypair, to authenticate. A shared key is then generated by performing the AKE protocol.

6.1 Post-Quantum OPAQUE implementation

Constructing a Post-Quantum version of the OPAQUE protocol requires the replacement of the used OPRF and AKE protocols with suitable post-quantum variants. We instantiate two Post-Quantum versions, one using our novel OPRF OPUS and the other one using our NR-OT OPRF. Both versions use a post-quantum secure replacement of the X3DH protocol, proposed by Hashimoto et al [HKKP21], as the AKE. We chose this AKE since it provides security against Key Compromise Impersonation (KCI) attacks and forward secrecy, as required by the OPAQUE protocol, and is suitable for implementation using CSIDH-based primitives. The protocol is based on a Key Encapsulation Mechanism (KEM) scheme and a signature scheme. We chose the CSIDH-based CSIKE [Qi22] as the KEM, since it is IND-CCA secure as required by the used AKE and is easy to implement. As the signature scheme, we chose CSI-FiSh [BKV19], as there already is an implementation available. The full protocol flow for the OPAQUE Password Registration and Password Authentication is detailed in Figure 11 and Figure 12 respectively. $\texttt{Ext}_\texttt{s}$ and $\texttt{F}_\texttt{K}$ are PRF using KMAC256 instead of HMAC256, since we require variable length output. The PRF uses `s` and `K` as the respective keys, with different labels to differentiate between $\texttt{Ext}_\texttt{s}$ and $\texttt{F}_\texttt{K}$.

Figure 11: Description of the Post-Quantum OPAQUE Password Registration



6.2 Comparison to Pre-Quantum implementation

To measure the performance difference, we compare our implementation to `libopaque`, an open-source, pre-quantum implementation of OPAQUE (`https://github.com/stef/libopaque`). The average execution time for the client and the server is shown in Table 3, while the communication cost is shown in Table 4. Our implementation is the first Post-Quantum secure instantiation of the OPAQUE protocol. While it leads to a increase in execution time and communication cost, this concretizes the overhead of switching to post-quantum cryptography for advanced protocols.

7 Case Study: Private Set Intersection

In a private set intersection (PSI), two or more parties, commonly a server and a client, hold data sets S and C. After performing the PSI protocol, one or both parties learn $S \cap C$ without revealing anything about the other parties set. In the client-server case, the sets are very often *unbalanced*, as the server set is

Figure 12: Description of the Post-Quantum OPAQUE Password Authentication and Key Generation

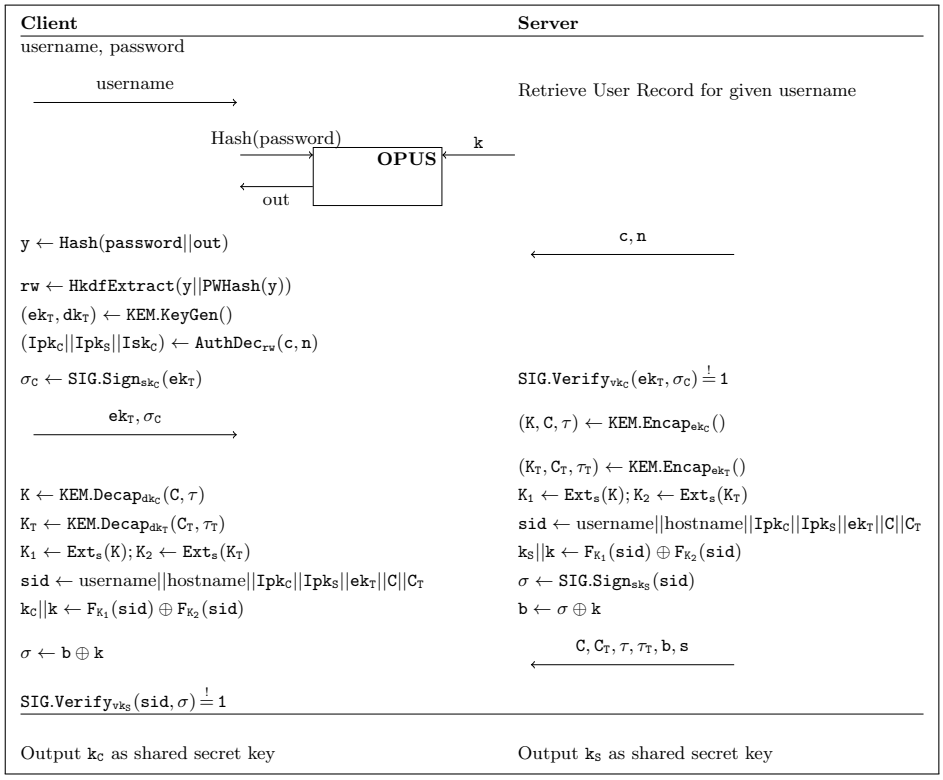


Table 3: Comparison between the execution time of libopaque and our two Post-Quantum OPAQUE instantiations. The execution time is averaged over 100 runs.

Function	libopaque	PQ		PQ / libopaque	
		OPUS	NR-OT	OPUS	NR-OT
Reg. Client	119.37ms	40.58s	11.24s	× 339.94	× 94.17
Reg. Server	95.63ms	41.06s	11.74s	× 429.33	× 122.76
Auth. Client	96.54ms	32.16s	3.40s	× 333.17	× 35.22
Auth. Server	120.32ms	32.01s	2.91s	× 268.15	× 24.34

much larger than the client set $|S| \gg |C|$. A well-studied application of PSI is Private Contact Discovery, where clients want to know which of their contacts

Table 4: Comparison between the communication overhead of libopaque and our two Post-Quantum OPAQUE instantiations

Function	libopaque	PQ		PQ / libopaque	
		OPUS	NR-OT	OPUS	NR-OT
Reg. Client	224B	64kiB	817kiB	× 294.4	× 3733
Reg. Server	64B	48kiB	144kiB	× 770	× 2307.4
Auth. Client	160B	17kiB	769kiB	× 106.1	× 4920.2
Auth. Server	320B	65kiB	161kiB	× 208.2	× 515.7

also use the same service to communicate over the service. [KRS+19]

To perform PSI using OPRFs, the holder of the larger set computes the PRF for each set entry and, optionally, inserts the results in an efficient data structure, e.g. a cuckoo filter. Then, the OPRF is computed in the online phase. The client uses their set entries as input and the server oblivious evaluates them with the same key as in the keyed PRF and checks whether the result is in the filter or not.

Performing PSI without a verifiable OPRF may lead to a tagging attack where a malicious server uses different keys for each client when performing the OPRF, leading to the identification of the results later. This is why previous work by [KRS+19] relaxes the security assumption and assumes a malicious client and a semi-honest server. They also postulate three goals for unbalanced PSI: The server should perform the computationally most expensive tasks, all expensive tasks are only performed once and updates are fast. We now instantiate their PSI framework with both isogeny-based OPRFs and compare it to our implementation. Of independent interest, we propose a small optimization for the setup of the elliptic curve Naor-Reingold(ECNR) PSI protocol in Appendix A. The results can be found in Table 5.

7.1 PSI with ECNR

The ECNR-PSI protocol is divided into three phases: First setup phase, where a Cuckoo filter is filled with the PRF results of server set entries and sent to the client. Then, a base phase, where some initial, data-independent Oblivious Transfer is performed. Using cheap symmetric cryptography, the parties generate many more OT pairs from this base OT using a technique called *OT Extension*. Then, in the online phase, the OPRF is performed using the extended OT pairs. This is currently the most efficient PSI protocol. [KRS+19]

7.2 PSI with NR-OT

The implementation with the NR-OT is relatively close to the ECNR files. The setup phase is identical other than replacing the communication interface with the one provided by the PQ-OT implementation. Since the PQ-OT implementation does not provide an implementation for OT extensions, we skip the base

phase and only implement an online phase. In the online phase, the OPRF is performed with all client elements.

Table 5: PSI comparison using ECNR, NR-OT, and OPUS as the OPRF for set intersection. The ECNR column combines base and online for better comparability.

	parameters		setup		online	
	$\|S\|$	$\|C\|$	$\|S\|$	$\|C\|$	$\|S\|$	$\|C\|$
NR-OT	2^0	2^0	0.28s	0.55s	1.13s	0.84s
			134 bytes	1 byte	128 kiB	0.75MiB
	2^5	2^5	1.66s	1.97s	4.17s	3.95s
			263 bytes	1 byte	4MiB	8.5 MiB
	2^{10}	2^{10}	46.85s	47.12s	105.22s	105s
			4.31 MiB	1 byte	128 MiB	256.6 MiB
OPUS	2^0	2^0	0.39s	0.37s	16.60s	17.16s
			133 bytes	0 bytes	17.07 kiB	9.04 kiB
	2^5	2^5	10.00s	10.00s	486.14s	486.55s
			262 bytes	0 bytes	546.25 kiB	290.26 kiB
	2^{10}	2^{10}	303.38s	303.38s	16367.12s	16367.60s
			4.31 kiB	0 bytes	34.14 MiB	18.08 MiB
ECNR	2^0	2^0	0.01s	0s	0.23s	0.05s
			133 bytes	0 bytes	12.04 kiB	16 bytes
	2^5	2^5	0.02s	0s	0.21s	0.06s
			262 bytes	0 bytes	137.05 kiB	512 bytes
	2^{10}	2^{10}	0.3s	0s	0.64s	0.57s
			4.36 kiB	0 bytes	4.04 MiB	16 kiB

The communication overhead may be lower when using OT extensions, which uses symmetric cryptography to generate more OT pairs from a few base OT queries. [BDK+20] show that the IKNP protocol [IKNP03] is secure against quantum adversaries conditional on updating the bit length of both the hash function and the base OT length, but unfortunately do not integrate the extensions in their implementation.

7.3 PSI with OPUS

To perform PSI with OPUS, several observations are important to obtain an efficient protocol: First, the protocol is relatively stateless. The client has to keep no state about where it is in the execution, as only one message is in transit at any given time, which can be identified with an ID. Therefore, the only state kept about an element in transit is a corresponding unblinding key.

The server has to take an additional precautionary measure to keep the client from leaking intermediate evaluations. For this, the server pre-generates

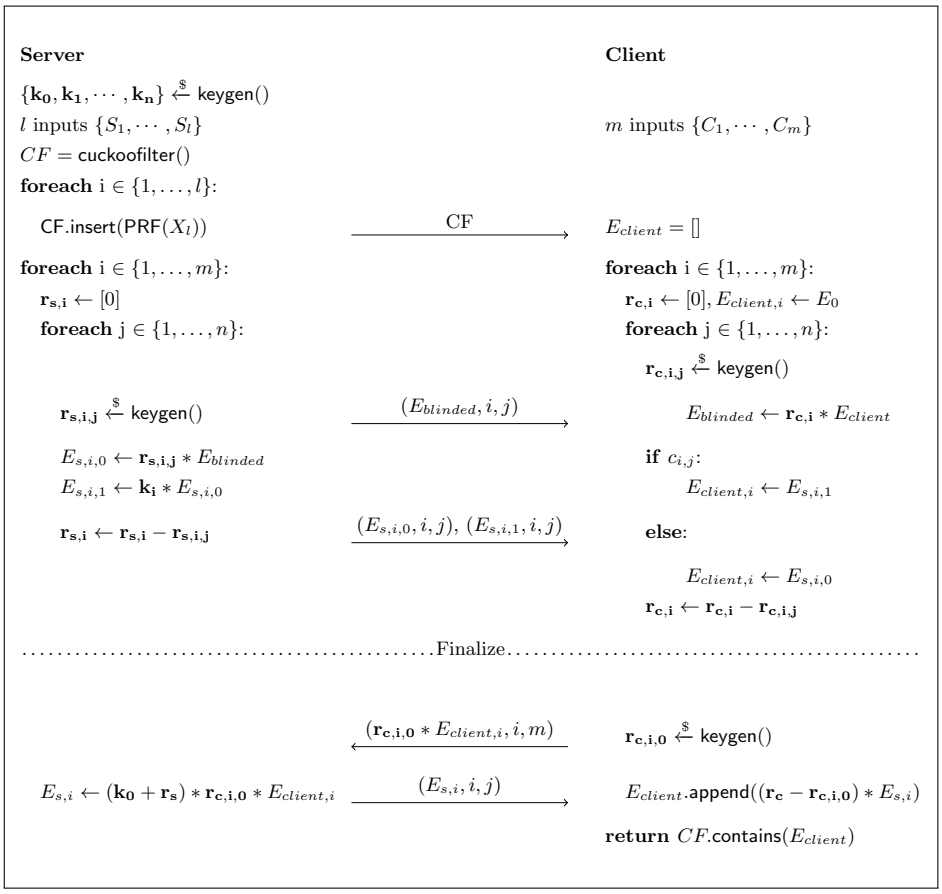


Figure 13: Amortizing the round cost of OPUS by reducing the state and adding labels.

all blinding keys and computes the unblinding element at the time an element is first seen.

In our measurements, the client seems to perform badly in the setup phase. This is a measurement artifact as most of the time is spent waiting for the cuckoo filter from the server due to the choice of network connection.

7.3.1 Updatable OPRF

For very large sets, the probability that several elements are quite similar is quite good. It would be nice to take an existing evaluation and update the value where the bits differ. This could yield a runtime improvement: consider two inputs X_1, X_2 and the evaluation $Y_1 = \mathsf{OPUS}(X_1)$, with $X_1 \oplus X_2$ having a

low Hamming weight. A potential improvement could come from an updatable form of OPUS, where Y_1 is updated at the indices. For example, imagine X_1 and X_2 only differ at the first bit, which is set in X_2 but not X_1, and the third bit, which is not set in X_2 but is set in X_1. Then, $\mathsf{OPUS}(X_2)$ can be computed as $\mathsf{OPUS}(X_1) = k_1 * k_3^{-1} * \mathsf{OPUS}(X_2)$. This results directly from the commutativity of CSIDH.

The simple realization of this functionality has the client reveal 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 $(\mathbf{r} * \mathbf{k_i}^{-1} * E, \mathbf{k_i} * \mathbf{r} * E)$ for the given indices i. The client iteratively updates the PRF by selecting the correct output. Note that the finalization step is still necessary for the unblinding to ensure that no intermediate results are leaked, but without adding $\mathbf{k_0}$.

While this produces another PRF result, the simple protocol violates the OPRF security guarantee of the server learning nothing about the client input, since the server knows the index where two evaluations differ. An extended version sends some dummy indices as well and requires the server to respond with $(\mathbf{r} * \mathbf{k}^{-1} * E, \mathbf{r} * E, \mathbf{k} * \mathbf{r} * E)$, with $\mathbf{r} * E$ being used if the index was a dummy index. This approach would reduce the latency introduced by the rounds and the group actions, but requires either very similar inputs or extensive preprocessing by the client to ensure the results are updated ideally.

7.4 Result and Overhead

We compare against the EC-NR implementation of [KRS+19] as it is the most performant implementation of OPRFs for set intersection. We point out that recent work [HSW23] optimizes the PSI protocol with sublinear communication size of the server's client database, which may make the ECNR protocol more efficient.

8 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 9 and provide parameter-agnostic approximations in Table 8, 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, 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, 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] 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 6 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, the NR-OT OPRF 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. The current impelmentation gives around 14.4s of execution time, making the NR-OPRF with a CSIDH security parameter $p = 512$ vastly faster. However, the authors describe an optimization that could lead to both OPRFs matching in speed.

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 6: 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	verifiable
[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	16.9MB	●-◐	✓	✓	✓	✓
[Bas23]	Isogenies $\mathbb{F}_{p^2}$	2	3.0MB	●-●	✓	✗	✗	✗
[Bas23]	Isogenies $\mathbb{F}_{p^2}$	2	8.7MB	●-●	✓	✗	✗	✓
NR-OT	Isogenies $\mathbb{F}_p$ + lattices	2	20.54 kB	◐-◐	✓	✗	✗	✗
NR-OT	Isogenies $\mathbb{F}_p$ + lattices	4	34.88 kB	●-◐	✓	✗	✗	✗
NR-OT	Isogenies $\mathbb{F}_p$ + lattices + HE OT	2	640 kB	◐-◐	✓	✓	✓	✗
OPUS	CSIDH	258	24.7 kB	◐-◐	✓	✓	✓	✗

9 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 explores the generic construction of Naor-Reingold protocols, which traditionally use oblivious transfer to send blinded private keys. 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 a 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 9. 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 code to obtain our benchmarks and the CSV files for the figures are available with at `https://github.com/meyira/OIDA`.

To show the real-world impact of our protocols, we benchmarked the OPRFS for two use-cases: first, asymmetric password authentication using OPAQUE, where we report an overhead of around 35× for authentication and 123× for registration. To the best of our knowledge, this is the first implementation of a post-quantum version of OPAQUE. Second, we implement private set intersection with the OPRFs. To the best of our knowledge, this is the first implementation of PSI using isogenies.

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 and the NR-OT OPRF, especially in settings with low bandwidth.

The recent call for threshold cryptography by NIST [3] opens a new avenue for post-quantum threshold schemes which distribute the secret key amongst several servers but only requires that 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 and the NR-OT to obtain a threshold OPRF.

On an implementation side, we point out that the current implementations are neither optimized nor side-channel free, and that the code is not audited. A side-channel free implementation should be relatively easy for OPUS, as it only requires side-channel free key addition and group actions, as well as the conditional assignment of E_{client}. 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 OPUS using only isogenies.

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 10, Piotr Dobrowolski for ensuring the benchmarks are done in time for the submission and Yifan Zheng for spotting two errors in an earlier draft of this paper. We thank the authors of [BKW20] for confirming our suspicions about the protocol and clarifying their instantiation.

Lena Heimberger, Sebastian Ramacher and Christian Rechberger received funding from the Digital Europe Program under grant agreement number 101091642 ("QCI-CAT"). Fredrik Meisingseth was supported by the "DDAI" COMET Module within the COMET – Competence Centers for Excellent Technologies Programme, funded by the Austrian Federal Ministry (BMK and BMDW), the Austrian Research Promotion Agency (FFG), the province of Styria (SFG) and partners from industry and academia. The COMET Programme is managed by FFG.

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 func-

[3] `https://csrc.nist.gov/projects/threshold-cryptography/`

tions 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.

[BDK+20] Niklas Büscher, Daniel Demmler, Nikolaos P. Karvelas, Stefan Katzenbeisser, Juliane Krämer, Deevashwer Rathee, Thomas Schneider, and Patrick Struck. Secure two-party computation in a quantum world. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, ACNS 20, Part I, volume 12146 of LNCS, pages 461–480. Springer, Heidelberg, October 2020.

[BFGP23] Ward Beullens, Luca De Feo, Steven D. Galbraith, and Christophe Petit. Proving knowledge of isogenies – a survey. Cryptology ePrint Archive, Paper 2023/671, 2023. https://eprint.iacr.org/2023/671.

[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.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, CRYPTO 2012, volume 7417 of LNCS, pages 868–886. Springer, Heidelberg, August 2012.

[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.

[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.

[DFK+23] Luca De Feo, Tako Boris Fouotsa, Péter Kutas, Antonin Leroux, Simon-Philipp Merz, Lorenz Panny, and Benjamin Wesolowski. SCALLOP: Scaling the CSI-FiSh. In PKC 2023, Part I, LNCS, pages 345–375. Springer, Heidelberg, May 2023.

[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.

[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.

[ECS+15] Adam Everspaugh, Rahul Chatterjee, Samuel Scott, Ari Juels, and Thomas Ristenpart. The pythia PRF service. In Jaeyeon Jung and Thorsten Holz, editors, USENIX Security 2015, pages 547–562. USENIX Association, August 2015.

[EKP20] Ali El Kaafarani, Shuichi Katsumata, and Federico Pintore. Lossy CSI-FiSh: Efficient signature scheme with tight reduction to decisional CSIDH-512. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, PKC 2020, Part II, volume 12111 of LNCS, pages 157–186. Springer, Heidelberg, May 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.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. https://eprint.iacr.org/2012/144.

[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.

[HKKP21] Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski, and Thomas Prest. An efficient and generic construction for signal's handshake (x3dh): Post-quantum, state leakage secure, and deniable. Cryptology ePrint Archive, Paper 2021/616, 2021. https://eprint.iacr.org/2021/616.

[HSW23] Laura Hetz, Thomas Schneider, and Christian Weinert. Scaling mobile private contact discovery to billions of users. Cryptology ePrint Archive, Paper 2023/758, 2023. https://eprint.iacr.org/2023/758.

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

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, CRYPTO 2003, volume 2729 of LNCS, pages 145–161. Springer, Heidelberg, August 2003.

[JKX18] Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu. OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks. In Jesper Buus Nielsen and Vincent Rijmen, editors, EUROCRYPT 2018, Part III, volume 10822 of LNCS, pages 456–486. Springer, Heidelberg, April / May 2018.

[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 Delpech 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.

[Qi22] Mingping Qi. An efficient post-quantum kem from csidh. Journal of Mathematical Cryptology, 16(1):103–113, 2022.

[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`.

[SEA21] Microsoft SEAL (release 3.7). `https://github.com/Microsoft/SEAL`, September 2021. Microsoft Research, Redmond, WA.

[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 Improving the PSI setup phase for the ECNR PRF

The setup phase of the ECNR-PRF in the PSI protocol in Section 7 is quite slow, as e.g. a PRF with 128 bits of input requires, on average, 65 point multiplications on the elliptic curve. We propose using lookup tables to speed up the computations.

The lookup table (LUT) contains a set of x bits where all possible combinations of multiplication results per set bit are stored. More concretely, the input space is split into blocks of size x during a pre-computation phase. Within those blocks, all possible combinations of input values are precomputed. That means in the case of $x = 4$ and a block with $\{x_1, \ldots, x_4\}$, the LUT will consider all possible inputs $\{0001, \ldots, 1111\}$, and set the table entries to the corresponding intermediate NR values $\{k_4, \ldots, k_1 * k_2 * k_3 * k_4\}$. To compute the PRF, the correct entry for an input block is retrieved from the LUT and multiplied with the other input blocks. Using this optimization, we can get a performance improvement for the PRF computation of up to almost 16%.

LUT size	nr. mult.	pre-comp. in ms	Time for PRF comp.	improv. with LUT	improv. w/o LUT
2^0 -1	2^7	/	599.45s	/	/
2^2 -1	2^6	0.05ms	572.10s	4.78%	4.78%
2^4 -1	2^5	0.16ms	538.26s	11.36%	11.37%
2^8 -1	2^4	6.24ms	514.61s	16.48 %	16.49%
2^{16}-1	2^3	1.73s	482.11s	17.74 %	18.15%

Table 7: Performance differences when computing 2^{21} ECNR PRFs with lookup tables of differing size. The performance improvement is given with and without taking LUT generation into account. Nr.mult. gives the number of required multiplications to obtain a PRF. The LUT size is computed with -1 as the all-zero entry is disregarded.

Note that the lookup table for a specific block grows exponentially in size. We split the blocks evenly and get the best result at a block size of 16 bits. After this point, the memory complexity becomes too large. Potential fine-tuning with unbalanced blocks may yield even faster results (e.g. splitting 128-bit input in blocks of 20 and a final block of 8 results in seven blocks that need to be multiplied instead of eight), but we leave this optimization for future work as we believe the gain to be negligible. The corresponding code can be found in `code/mobile_psi_cpp`.

B Performance

To compare the different protocols for computing the Naor-Reingold OPRF from isogenies, we give a cost overview in Table 8. 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

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

The comparison (Table 8) between the NR-OT protocol from Section 2.3 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 8: 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.

protocol	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 9, we show how the communication complexity compares for different parameter sets 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 efficient OPRF that is secure against semi-honest adversaries in a post-quantum setting.

Table 9: Communication and computation complexity estimates of the protocols with different concrete parameters.

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	◐-◐

C Freely and Transitively Acting Class Group

We revisit the necessary properties of the class group action for our security proofs. 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,

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

with I being the identity element of the group. This is important to capture all isomorphic curves. Further, the class group action is said to act *transitively*, if

$$\forall (x, y) \in Cl(\mathcal{O})^2 : gx = y$$

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 the bound $[-m, m]$. 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.

D OPUS Security Proof

For the security proof, we consider the one-more pseudorandomess security game of Everspaugh et al. [ECS+15] in the fully oblivious setting.

Definition 3. A OPRF $F_k : \mathcal{M} \to \mathcal{R}$ provides one-more pseudorandomess if for any PPT adversory $\mathcal{A}$ the advantage in the one-more pesudorandomness experiment defined in Figure 14, $|\Pr[\text{om-PRF} = 1] - \frac{1}{2}|$ is negligible.

Experiment om-PRF:

- $(pk, sk) \leftarrow\!\!\$\ \mathcal{K}$, $q, c \leftarrow 0$
- $(i_1, \ldots, i_\ell, b') \leftarrow \mathcal{A}^{\text{RoR,PRF-Srv}}$
- If $\ell > q$ or $c \geq \ell$ or $\exists \alpha \neq \beta : i_\alpha = i_\beta$ return 0.
- Return $\beta' = \bigoplus_{\alpha=1}^{\ell} b_{i_\alpha}$

RoR(m):

- $q \leftarrow q + 1$, $b_q \leftarrow\!\!\$\ \{0, 1\}$
- $Z_0 \leftarrow\!\!\$\ \mathcal{R}$, $Z_1 \leftarrow F_k(m)$
- Return Z_{b_q}

PRF-Srv(m):

- $c \leftarrow c + 1$
- Return PRF-Srv$_k(m)$

Figure 14: Security game for one-more pseudorandomness.

This notion, as shown by Everspaugh et al., implies the weaker one-more unpredictability security notion of OPRFs. Note though, that in Figure 14, the PRF-Srv oracle is modelled as a single query. In our case, this algorithm takes part in a multi-round protocol, whereas the output depends on client-provided random values which on their own depend on previous outputs of PRF-Srv. We will however keep the notation for simplicity and assume that all the required information to produce a transcript is passed as part of m.

We now show that OPUS is one-more pseudorandom based on the D-CSIDH assumption:

Theorem 1. *If the D-CSIDH assumption holds, then OPUS is one-more pseudorandom.*

Proof. The basic idea is to replace the use of the secret key k_i step-by-step with randomly sampled curves.

- Game 0: The initial game.
- Game i: Everything is as before, but compute $E_{s,i,1}$ by sampling uniformly at random from $\mathcal{E}$.
- Transition $i-1$ to i: an adversary that can distinguish between game $i-1$ and i, can also solve D-CSIDH. Indeed, let (E, H, E', H') be from a D-CSIDH challenger. We set $E_{s,i,0} \leftarrow H$ and $E_{s,i,1} \leftarrow H'$ which interpolates between the two games.[4]

In Game n, the adversary can only guess as none of the $k_1, \dots k_n$ are used in the protocol execution. □

Proofing the security of OPUS in the universal composability model and in an adaptive setting, is currently open and future work. To achieve adaptve security, it would be required at least to produce the output of OPUS via a random oracle, i.e., by outputting $H(m, \mathcal{E}_c lient)$, as observed by Jarecki et al. [JKX18].

E Auxilary Figures

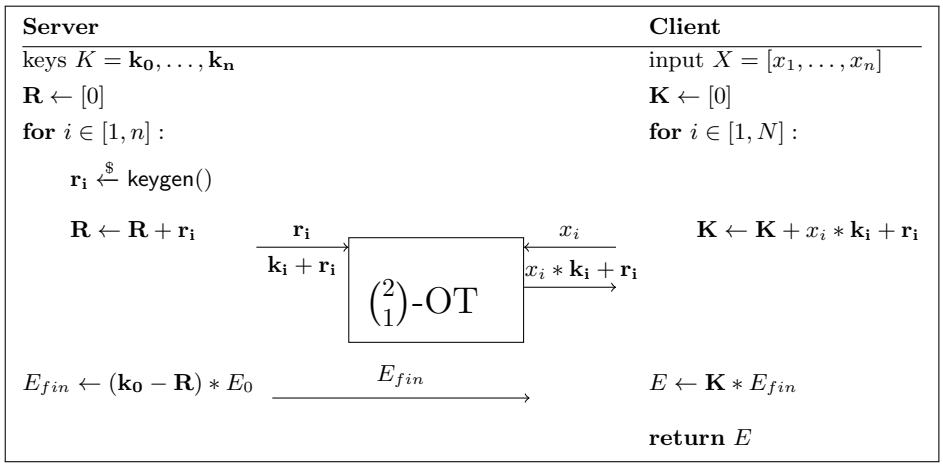


Figure 15: Evaluating the NR-OPRF with CSIDH and N OT calls.

[4] We could set $E_0 \leftarrow E$ and E' would represent the public key of the server. As we do not have a public key, though, this step is not required.

F Artifact Description

We provide all benchmark-generating files as well as example instantiations of OPUS in the artifact under `https://anonymous.4open.science/r/opus-artifact-E557/` 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/`.

F.1 Generation of Figure 4

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

F.2 Generation of Figure 5

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

F.3 Generation of Table 1

`opus.c` performs local computations of OPUS. The code was used to generate Table 2.

F.4 Generation of Figure 10

The online evaluation of OPUS with different servers in London in Figure 10 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
`code/opus/files/simple-client.c`. The server is multithreaded, the source is in `code/opus/files/simple-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.

F.5 OPUS-PSI

The files for OPUS are in a seperate folder in `code/opus-psi`. This is practically a stripped-down version of the Contact Discovery Impelmentation [KRS+19]

F.6 Approximation for Key Leakage

`code/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 3.2.1.

F.7 Generic Naor-Reingold OPRF (NR-OT)

We cloned the PQ-MPC implementation from [BDK+20] and In `code/nr-ot/test/oprf_server.cpp` and
`code/nr-ot/test/oprf_client.cpp`, we provide the code for the NR-OPRF using the PQ-OT from [BDK+20] and CSI-FiSh from [BKV19]. The PSI files `code/nr-ot/test/psi_server.cpp` and
`code/nr-ot/test/psi_client.cpp` can be found in the same folder, as well as the file `code/nr-ot/test/prf_opt.cpp` used to generate Figure 7. The CSV file of the measurements is in
`csv-files/prf_csifish.csv`

F.8 OPAQUE

The OPAQUE code is in `code/opaque` and includes both libopaque and our implementations. There were several changes to the used libraries:

- OPAQUE/X3DH
  - Ext_s and F_K do not use HMAC256 but instead KMAC256 to allow for variable length output
  - The used KMAC256 function does also not use a domain separator as described in the X3DH paper but instead uses a different customization string.
  - For the password used for encrypting the payload stored on the server, we truncate the output of the SHA512 output and only take the first 256 bits
  - Identifier for sid used as input to F_K functions uses the username and the hostname instead of the hash of long-term public keys as described OPAQUE Draft.
- TLS-OPAQUE
  - Server requires a certificate to sign key share, changed to accept almost all ciphers (only PSK should work without certificate).
- SEAL

  - `src/seal/util/locks.h` to add #include ¡mutex¿
  - `CMakeLists.txt` to compile library as SHARED and not STATIC
- emp-tool
  - NetIO: add a constructor for an already opened socket.
  - NetIO: remove closing of the socket from the destructor.
  - `CMakeLists.txt` to compile library as SHARED
- PQ-MPC
  - `CMakeLists.txt` adapt C++ standard from C++14 to C++17
  - `CMakeLists.txt` to compile library as SHARED