Efficient Distributed Oblivious Pseudorandom Functions from LWE

Helen Propson
MIT

May 10, 2023

Abstract

The ability to synthesize DNA easily and inexpensively has raised concerns about the potential misuse of this technology for creating harmful pathogens. Such bioweapons could cause global pandemics if accidentally or intentionally released. This paper proposes a cryptographic protocol to screen DNA synthesis orders against a database of known hazardous sequences to prevent the synthesis of dangerous biological agents. The screening system must aim to maintain the privacy of DNA synthesis queries while also keeping the known hazardous sequences confidential. We propose a protocol construction that utilizes a distributed oblivious pseudorandom function (OPRF) on the DNA synthesis queries. We then evaluate the construction using our definitions of correctness and security. The proposed protocol achieves semantic security against semi-honest adversaries under the LWE assumption, preserving the privacy of the DNA synthesis query without compromising the secrecy of the sequences stored in the database. As such, the protocol can be used to contribute to preventing the spread of access to dangerous bioweapons.

1 Introduction

As the ability to synthesize DNA [1] becomes more widespread, it becomes increasingly important to address the resulting risk that dangerous biological agents can be assembled. To mitigate this risk, a screening system is needed to check all DNA synthesis orders against a database of known hazardous biological agents. However, we want this system to work without revealing the dangerous sequences the orders are screened against. Furthermore, the privacy of the DNA synthesis query must also be maintained. It is also important that this screening process does not slow down ongoing DNA research.

The focus of this paper is to advance the development of such a cryptographic screening protocol, meeting all its security goals as set out by our Main Theorem (Theorem 4.1). To achieve this, we propose a construction of a distributed oblivious pseudorandom function (OPRF) to be used on the DNA synthesis queries and database entries (Section 3) and prove its security under the LWE assumption (Section 4). Using this protocol, a researcher wishing to synthesize a sequence of DNA can send their DNA synthesis order to multiple servers whose responses will allow the researcher to compute the encryption (the PRF) of the DNA synthesis order. This allows the researcher to query the database of encrypted hazardous sequences to find out if their synthesis order matches any of the database entries.With these elements in place, such a protocol will contribute to preventing the spread of access to dangerous bioweapons.

[1]here DNA synthesize refers to the process of artificially creating DNA molecules

1.1 Previous Work

As previously mentioned, the goal of this paper is to develop a screening protocol, which we approach by using a distributed oblivious pseudorandom function (OPRF). Concretely, a pseudorandom function (PRF) (which we define more formally in Definition 2.4 in Section 2.2) is an efficiently computable function[2] that takes in a secret key K and input x whose output cannot be distinguished from random in polynomial time [GGM86], thus making it a powerful cryptographic primitive[3]. An *oblivious* PRF (OPRF) is a two-party protocol between a client with input x and a server with secret PRF key K such that, at the end of the protocol, the client learns the PRF of its input and the server learns nothing [FIPR05]. There are weaker definitions of OPRFs that relax the limitations on what the server and client are allowed to learn.

Multiple variants of PRFs have been presented in the past. Two variants of particular relevance are distributed OPRFs and verifiable OPRFs. Distributed PRFs involve splitting the secret key amongst multiple servers that each compute a partial evaluation of the PRF [NPR99]. These partial evaluations can then be combined to get the PRF of the client's input. Verifiable OPRFs are OPRFs with the added guarantee to the client that the PRF they receive is indeed the PRF of the client's input using the server's secret key [ADDS21]. The client could receive this verification via a zero-knowledge proof provided by the server.

The extra guarantees provided by distributed and verifiable OPRFs are useful but not essential for the DNA screening application. While a distributed OPRF could increase security by requiring collusion of multiple servers to compromise the PRF, a single server could still be utilized instead. Furthermore, although a verifiable OPRF may not be necessary for the application, it could provide additional assurance to the client that the PRF they receive is valid.

1.2 Our Contribution

Our work differs from previous VOPRF constructions in two main ways. First, we relax the constraint that the protocol must be verifiable. Second, we adapt this protocol to be distributed to increase its resilience against a compromise in the secrecy of the key. Our protocol achieves semantic security, meaning that given two client inputs and a PRFs on one of the client's inputs, a semi-honest[4] server cannot distinguish which of the two inputs was used to produce the PRF. This property is crucial for preserving the privacy of the DNA synthesis query.

An additional area of interest is adapting the protocol to defend against adversaries with both quantum computation power as well as classical computation power. Future work may also focus on achieving both verifiability and a high level of efficiency.

1.3 Outline

The remainder of this paper is divided into 3 main sections. Section 2 begins by defining a Distributed Oblivious Pseudorandom Function, which is utilized by the protocol proposed later in the paper. It then outlines security definitions against which the protocol will be evaluated in the results section of the paper. Section 3 proposes a protocol construction. Finally, Section 4 is a proof of the techniques used to meet the definitions of correctness and security previously defined. The paper concludes with a discussion of the implications of our work.

[2]An efficiently commutable function is one that can be run in polynomial time

[3]A cryptographic primative is a low-level cryptographic algorithm used as a basic building block for higher-level cryptographic algorithms [oST20]

[4]A semi-honest server is a server that must participate in the protocol according to its specification

2 Distributed OPRF: Definition

2.1 Syntax and Correctness

Informally, an oblivious pseudorandom function (OPRF) is used in the context of two parties, Alice and Bob. Alice has an input x and wants the evaluation of PRF with key K on her input. Bob runs Gen to get a key K. They then run the protocol Π, after which Alice has the result of Eval, i.e. the evaluation of the PRF on Alice's input.

Definition 2.1. *An oblivious pseudorandom function (OPRF) is a pair of algorithms* $(\mathsf{Gen}, \mathsf{Eval})$ *together with a two-party protocol* Π *that has the following syntax:*

- $\mathsf{Gen}(1^\lambda)$ *is a probabilistic algorithm that outputs a key* K*;*
- $\mathsf{Eval}(K, x)$ *is a deterministic algorithm that, on input the key* K *and a string* $x \in \{0,1\}^\ell$*, outputs a string* $y \in \{0,1\}^m$*;*
- Π *is a protocol between a server that holds the key* K *and a client that holds an input* x*.*

The correctness property of an OPRF requires that Π *implements the functionality that maps* $(K, x) \to (\perp, \mathsf{Eval}(K, x))$*. That is, when party* B *(resp.* A*) starts with input* K *(resp.* x*), at the end of the protocol, party* A *outputs* $y = \mathsf{Eval}(K, x)$ *while party* B *outputs* $\perp$*.*

In the case that the server with key K (Bob) is compromised, the secret key K is leaked. As such, to make a stronger function, the key K can be distributed amongst n servers. The Gen algorithm is modified to output K together with $k_1, \ldots, k_n$ and Π becomes a protocol between $n+1$ parties. This is called a distributed oblivious pseudorandom function (DOPRF), defined formally below.

Definition 2.2. *An* n*-party distributed oblivious pseudorandom function (DOPRF) is the same as an OPRF except that:*

- $\mathsf{Gen}(1^\lambda)$ *is a probabilistic algorithm that outputs key* K *and* $k_1, \ldots, k_n$
- Π *is a protocol between* n *servers, each of which holds a share* k_i*, and a client that holds an input* x*.*

The correctness property of a DOPRF requires that Π *implements the functionality that maps*

$$(k_1, k_2, \ldots, k_n, x) \to (\perp, \ldots, \perp, \mathsf{Eval}(K, x)) \ .$$

For the purposes of this paper, we will focus on a protocol Π that consists of two interactions between the client and servers with the following form:

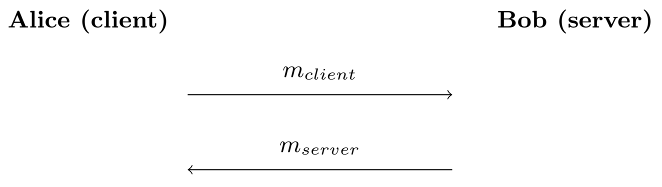

Figure 1: Two-interaction protocol format

2.2 Security

In addition to defining correctness, we must also define security for our protocol. It is important to note that we define security in the case of a semi-honest server and a semi-honest client. This means that we only consider the setting in which the server and client participate in the protocol according to its specification. Our security definition has two parts: security against a semi-honest server (obliviousness) and security against a semi-honest client (pseudo-randomness). We will start by defining security against a semi-honest server.

Obliviousness: Semi-honest Server. Informally, we want the semi-honest server to be oblivious to the client's input after participating in the protocol. Specifically, consider the scenario in which an adversary A (i.e. the semi-honest server) presents two possible inputs to a challenger C. C randomly selects one of the inputs and generates a protocol transcript with it, sending m_{client} back to A. The goal of the adversary is to correctly identify which input was chosen by the challenger. Obliviousness is achieved if the adversary cannot identify the correct input with non-negligible probability. Formally, we define the game as follows:

Definition 2.3. *Let λ be the security parameter. Let A and C be players in the game depicted in Figure 2.3. A sends two strings x_0 and x_1 to C. C draws a bit b from random and does reveal b to A. C uses b to send the message $m_{client}(x_b)$ to A and A returns bit b'.*

1. *A wins if b = b'*
2. *We say the protocol is oblivious if for all probabilistic polynomial-time adversaries A, pr[A wins] $\leq \frac{1}{2} + 2^{-\Omega(\lambda)}$*

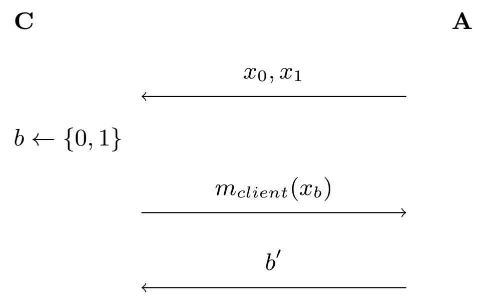

Figure 2: Obliviousness Definition 2.3

Pseudo-randomness: Semi-honest Client. To test for security against a semi-honest client, an adversary A (i.e. a semi-honest client) is challenged to distinguish a PRF from random. In this scenario, the adversary A engages in an interactive game with a challenger C. The adversary A obtains a polynomial number of PRF values paired with their corresponding evaluation transcripts from a transcript oracle PRF_k. The objective of the adversary A is to distinguish between another input-output pair of the PRF and random with a non-negligible probability. A formal definition is provided below.

Definition 2.4. *Psuedorandom Function [KL07]. An efficient, length-preserving, keyed function $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ is a psuedorandom function if for all probabilistic polynomial-time adversaries A, there exists a negligible function negligible such that*

$$\left| Pr[A^{F_k(\cdot)}(1^\ell) = 1] - Pr[A^{f(\cdot)}(1^\ell) = 1] \right| \leq negligible(\ell)$$

where $k \leftarrow \{0,1\}^\ell$ is chosen uniformly at random and f is chosen uniformly at random from the set of functions mapping ℓ-bit strings to ℓ-bit strings.

3 Distributed OPRF Protocol

3.1 Preliminaries

Our protocol construction utilizes the learning with errors (LWE) problem, first introduced in its current form in a seminal work by Oded Regev. The LWE problem asks to solve a system of noisy linear equations [Reg05]. That is, it asks to find $s \in \mathbb{Z}_q^\ell$ given

$$\{(a_i, \langle a_i, s\rangle + e_i) : s \leftarrow \mathbb{Z}_q^\ell, a_i \leftarrow \mathbb{Z}_q^\ell, e_i \leftarrow \chi\}_{i=1}^m$$

where

- $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ denotes the finite ring of integers modulo q, $\mathbb{Z}_q^\ell$ denotes the vector space of dimension ℓ over $\mathbb{Z}_q$;
- χ is a probability distribution over $\mathbb{Z}$ which typically outputs "small" numbers, an example being the uniform distribution over an interval $[-B, \dots, B]$ where $B \ll q/2$; and
- $a \leftarrow \mathcal{D}$ denotes that a is chosen according to the finite probability distribution $\mathcal{D}$, $a \leftarrow S$ denotes that a is chosen uniformly at random from the (finite) set S.

Below is the LWE assumption, which we will use to prove our protocol's security.

Definition 3.1. *(Decision) Learning with Errors [Reg05]. Let $\ell = \ell(\lambda)$ and $q = q(\lambda)$ be integer parameters and $\chi = \chi(\lambda)$ be a distribution over $\mathbb{Z}$. The Learning with Errors (LWE) assumption $LWE_{\ell,q,\chi}$ states that for all $m = poly(\lambda)$ the following distributions are computationally indistinguishable:*

$$(A, As + e) \approx_c (A, u)$$

where $A \leftarrow \mathbb{Z}_q^{m\times\ell}, s \leftarrow \mathbb{Z}_q^\ell, e \leftarrow \chi^m, u \leftarrow \mathbb{Z}_q^m$.

Next, we have the Noise Flooding Lemma and Leftover Hash Lemma which we will also use to prove security.

Lemma 3.2. *Leftover Hash Lemma [ILL89] Let P be a probability distribution over $\mathbb{Z}_q^m$. The following two distributions have statistical distance at most ϵ as long as $H_\infty(P) \geq \ell\log(q) + 2\log(1/\epsilon)$:*

$$(A, Ae \bmod q) \approx (A, u)$$

where $A \leftarrow \mathbb{Z}_q^{m\times\ell}$ is uniformly random, $e \leftarrow \chi$ us drawn from the probability distribution P, and $u \leftarrow \mathbb{Z}_q^\ell$ is uniformly random. Here, $H_\infty(P)$ refers to the min-entropy of P (where min-entropy is defined below by Definition 3.3).

Definition 3.3. *Let X be a random variable. The min-entropy of X is defined as*

$$H_\infty(X) \equiv \min_{u\in U}\{-\log(\mathbb{P}[X = u])\} = -\log(\max_{u\in U}\mathbb{P}[X = u]) \ .$$

Lemma 3.4. *Noise Flooding Lemma [AJL+12]. Let $B = B(\lambda), B' = B'(\lambda) \in \mathbb{Z}$ be parameters and let $U([-B, B])$ be the uniform distribution over the integer interval $[-B, B]$. Then for any $e \in [-B', B']$, the statistical distance between $U([-B, B])$ and $U([-B, B]) + e$ is at most B'/B.*

In our security proof, we will also use the following definition of computational indistinguishability.

Definition 3.5. *Computational Indistinguishability [RP] Let $\{X_n\}_{n\in N}$ and $\{Y_n\}_{n\in N}$ be ensembles of probability distributions where X_n, Y_n are probability distributions ranging over $\{0,1\}^{\ell(n)}$ for some polynomial $\ell(\cdot)$. We say that $\{X_n\}_{n\in N}$ and $\{Y_n\}_{n\in N}$ are computationally indistinguishable (abbr. $\{X_n\}_{n\in N} \approx \{Y_n\}_{n\in N}$) if for all non-uniform PPT (probabilistic polynomial-time) D (called the "distinguisher"), there exists a negligible function $\varepsilon(n)$ such that for all $n \in N$*

$$| Pr[t \leftarrow X_n, D(t) = 1] - Pr[t \leftarrow Y_n, D(t) = 1] | < \varepsilon(n) \ .$$

Finally, we will use the following definition of Round in our protocol construction.

Definition 3.6.

$$\mathsf{Round}(v_i) = \begin{cases} 0, & \textit{if } v_i \leq q/2 \\ 1, & v_i \geq q/2 \end{cases}$$

3.2 The Protocol

Now that we have defined our preliminaries, we may now define our protocol. Here we outline the construction of the protocol.

First, we provide the Gen algorithm.

Definition 3.7. Gen *takes as input nothing and produces n keys $[k_1, \ldots, k_n]$ drawn uniformly from random from $\mathbb{Z}_q^{\ell}$, together with K, where $K = \sum_{i=1}^{n} k_i$.*

Next, we provide the Eval algorithm.

Definition 3.8. Eval *takes as input x and K and outputs*

$$PRF_K(x) = PRF_{k_1}(x) + \ldots + PRF_{k_n}(x)$$

where $PRF_k(x) = \mathsf{Round}(H(x) \cdot k)$ where $\mathsf{Round} : \mathbb{Z}_q \rightarrow \mathbb{Z}_p$ is the function defined in Definition 3.6 and $H(x)$ is a hash function applied to x that outputs a vector of length ℓ.

Finally, we give the Π protocol. In the first step of this protocol, the client sends:

$$m_{client} = \left\langle A, H(x) + A \cdot s + e, (A_1, A_1 \cdot s + e_1), \ldots, (A_w, A_w \cdot s + e_w) \right\rangle \tag{1}$$

where $H(x)$ is an $\ell \times 1$ vector, A is an $\ell \times \ell$ matrix, s is an $\ell \times 1$ secret vector drawn uniformly at random, and e is an $\ell \times 1$ vector of small errors. We choose an w large enough such that $w > \ell^5$. The hash function H acts as a random oracle on the secret x. m_{client} is an $\ell \times 1$ vector.
In the second interaction of Π, each of the n servers responds to the client's message (Equation 1) with:

$$\begin{aligned} m_{server_i} &= \left\langle k_i \cdot A + A'_i, k_i \cdot {m_{client}}^T + A'_i \cdot s + e'_i \right\rangle \\ &= \left\langle k_i \cdot A + A'_i, k_i \cdot (H(x) + A \cdot s + e)^T + A'_i \cdot s + e'_i \right\rangle \end{aligned} \tag{2}$$

where $\vec{k_i}$ is a $\ell \times 1$ vector from Gen, A'_i is

$$A'_i = \sum_{j=1}^{w} b_{ij} \cdot A_j$$

for $b_{ij} \leftarrow 0, 1$ drawn uniformly at random, e'_i is an $\ell \times 1$ vector of small errors given by

$$e'_i = \sum_{j=1}^{w} b_{ij} \cdot e_j \ .$$

Hence, $A'_i \cdot s + e'_i$ is a random linear combination of LWE samples given by

$$A'_i \cdot s + e'_i = \sum_{j=1}^{w} b_{ij} \cdot (A_j \cdot s + e_j) = (\sum_{j=1}^{w} b_{ij} \cdot A_j) \cdot s + \sum_{i=1}^{w} b_i \cdot e_j = A'_i \cdot s + e'_i$$

Therefore, m_{server} is an $\ell \times 1$ matrix.

The client can compute the partial PRF using the servers' responses (Equation 2):

$$\begin{aligned} v_i &= \Big(k_i \cdot (H(x) + A \cdot s + e) + A'_i \cdot s + e'_i\Big) - (k_i \cdot A + A'_i) \cdot s \\ &= k_i \cdot H(x) + (k_i \cdot A + A'_i) \cdot s + (k_i \cdot e + e'_i) - (k_i \cdot A + A'_i) \cdot s \\ &= k_i \cdot H(x) + (k_i \cdot e + e'_i) \end{aligned} \tag{3}$$

Then the client sums all of the partial PRFs (Equation 3) to produce

$$v = v_1 + \ldots + v_n \ ,$$

where $v \in \mathbb{Z}_q^\ell$ is a $\ell \times 1$ vector with each element $v_i \in \{0, 1, \ldots q-1\}$. The output is $\mathsf{Round}(v)$.

4 Results

In the following three subsections, we will be proving the correctness and security of the protocol given above. Formally, we will prove the following theorem.

Theorem 4.1. *Main Theorem. There exists a protocol Π described in Section 3.2 that satisfies the Correctness Definition 2.2 in Section 2.1 and under the LWE assumption, satisfies the Security Definitions 2.3 and 2.4 in Section 2.2.*

4.1 Correctness

In this section, we will prove Lemma 4.2 which says that our protocol satisfies the definition of correctness provided in Section 2.1. To do so, we prove that the evaluation of the PRF on the client's input x is in fact what the client receives at the end of the protocol.

Lemma 4.2. *There exists a protocol Π described in Section 3.2 that satisfies the Correctness Definition 2.2 in Section 2.1.*

Proof. The real evaluation of the PRF on the input is given by

$$PRF_K(x) = \mathsf{Round}(K^T H(x)) = \mathsf{Round}((k_1 + k_2 + \ldots + k_n)^T \cdot H(x)) \ . \tag{4}$$

The vector v computed by the client above is

$$v = (k_1 + k_2 + \ldots + k_n)^T \cdot H(x) + \varepsilon \tag{5}$$

where ε is an error term εB. The error term is computed as follows

$$\begin{aligned}\epsilon &= \sum_{i=1}^{n}(k_i \cdot e + e'_i) \\ &= \sum_{i=1}^{n}(k_i \cdot e + \sum_{j=1}^{w} b_{ij} \cdot e_j) \ .\end{aligned}$$

The vector v (Equation 5) is equivalent to the real PRF (Equation 4) in the event that each coordinate of v falls within the interval $v_i \in [0, q/2 - B] \cup [q/2 + B, q - 1]$ where B is an upper bound on the magnitude of the error term in v.

The probability that a coordinate does not fall within this interval is given by

$$\Pr[v_i \notin [0, q/2 - B] \cup [q/2 + B, q - 1]] \leq 2B/q \ ,$$

so by the union bound, the probability there exists a coordinate in v doesn't fall within the interval that gives no error is given by

$$\Pr[\exists v_i \in v \text{ where } v_i \notin [0, q/2 - B] \cup [q/2 + B, q - 1]] \leq \ell \cdot 2B/q \ ,$$

thus giving us an upper bound on the probability that the real PRF is not equivalent to the output of our protocol. Therefore we may pick a value of q that is large enough to give us the desired correctness probability. □

4.2 Obliviousness: Security against Semi-honest Server.

Here we show that our protocol meets the definition of obliviousness provided in Section 2.1. Specifically, we show that the semi-honest server cannot correctly identify which input the client chose to generate a protocol transcript with given two possible inputs.

Lemma 4.3. *There exists a protocol* Π *described in Section 3.2 that, under the LWE assumption, satisfies the Security Definition 2.3 in Section 2.2.*

Proof. We show this proof via a hybrid argument. We wish to show that the adversary cannot distinguish between a client's message generated using x_0 versus one generated using x_1, where x_0 and x_1 are adversarially generated inputs.

In the first hybrid, the server is sent

$$m_{client}(x_0) = \left\langle A, H(x_0) + A \cdot s + e, A_1 \cdot s + e_1, \ldots, A_w \cdot s + e_w \right\rangle \tag{6}$$

We can write (6) as

$$\begin{bmatrix} A \\ A_1 \\ \vdots \\ A_w \end{bmatrix} \cdot s + \begin{bmatrix} e \\ e_1 \\ \vdots \\ e_w \end{bmatrix} + \begin{bmatrix} H(x_0) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \ . \tag{7}$$

Hybrid 2 is the following distribution

$$\begin{bmatrix} u_1 \\ u_1 \\ \vdots \\ u_w \end{bmatrix} + \begin{bmatrix} H(x_0) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{8}$$

which is indistinguishable from (7) by the Learning with Errors (LWE) assumption. Hybrid 3 is the following distribution

$$\begin{bmatrix} u_1 \\ u_1 \\ \vdots \\ u_w \end{bmatrix} + \begin{bmatrix} H(x_1) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{9}$$

which is identically distributed to (8). The final hybrid is the following

$$\begin{bmatrix} A \\ A_1 \\ \vdots \\ A_w \end{bmatrix} \cdot s + \begin{bmatrix} e \\ e_1 \\ \vdots \\ e_w \end{bmatrix} + \begin{bmatrix} H(x_1) \\ 0 \\ \vdots \\ 0 \end{bmatrix} , \tag{10}$$

which is indistinguishable from (9), once again using the Learning with Errors (LWE) assumption.

The last hybrid (10) is the client's message

$$m_{client}(x_1) = \left\langle A, H(x_1) + A \cdot s + e, A_1 \cdot s + e_1, \ldots, A_w \cdot s + e_w \right\rangle$$

if the challenger chooses the other input, x_1. Therefore, the server is unable to distinguish between the two inputs, and our protocol is secure against semi-honest servers. □

4.3 Pseudo-randomness: Security against a Semi-honest Client

In this subsection, we will prove our protocol is secure against semi-honest semi-honest clients. In particular, we evaluate the protocol against the definition of pseudo randomness given in Section 2.1 which stated that the client should not be able to distinguish between the actual evaluation of PRF with key K on the client's input and the evaluation of a random function f. Since the servers' responses are generated using a secret key $\vec{k_i}$ that is not revealed to the client, to prove client security, we must show that the servers' responses do not reveal anything to the client except for the evaluation of the PRF of the client's input, as given by the definition in Section 2.2.

Lemma 4.4. *There exists a protocol* Π *described in Section 3.2 that satisfies the Security Definition 2.4 in Section 2.2.*

Proof. In the second interaction of π, the client receives

$$m_{server_i} = \left\langle k_i \cdot A + A_i', k_i \cdot (H(x) + A \cdot s + e)^T + A_i' \cdot s + e_i' \right\rangle$$

which can be written as

$$(k_i \cdot A + A_i') \cdot s + (k_i \cdot e + e_i') + k_i \cdot H(x)$$

We first note that $(k_i \cdot A + A_i')$ is indistinguishable from a random matrix A_i'' by the Leftover Hash Lemma (Lemma 3.2).

For the second step in this proof, we note that the second term in the server's response, $(k_i \cdot e + e_i')$, is indistinguishable from a random e_i'' by the Noise Flooding Lemma (Lemma 3.4).

Putting the first and second steps together, we can rewrite the server's message as

$$A_i'' \cdot s + e_i'' + k_i \cdot H(x) \ .$$

Summing these responses together is thus equivalent to

$$\mathsf{Eval}(K, x) + \text{random noise} .$$

Therefore, the client only learns the evaluation of the PRF on the input, and our protocol is secure against a semi-honest semi-honest client. □

We conclude our proof of Theorem 4.1 by noting that Lemma 4.2, Lemma 4.3, and Lemma 4.4 together prove Theorem 4.1 by proving the correctness, obliviousness, and pseudorandomness of our protocol, respectively.

5 Conclusion

In conclusion, this paper presents a cryptographic protocol that utilizes a distributed oblivious pseudorandom function (OPRF) to screen DNA synthesis orders against a database of known hazardous sequences to prevent the synthesis of dangerous biological agents. Our work differs from previous VOPRF constructions by adapting the protocol to be distributed to increase its resilience against a compromise in the secrecy of the key, and by relaxing the constraint that the protocol must be verifiable. The protocol meets the security goals set out by the Main Theorem 4.1 and achieves semantic security against semi-honest adversaries, preserving the privacy of the DNA synthesis query without compromising the secrecy of the sequences stored in the database. This protocol contributes to preventing the spread of access to dangerous bioweapons, and future work may focus on achieving both verifiability and efficiency. Verifiability would strengthen security in the case that the server is not required to be semi-honest. Overall, this paper presents an important step forward in addressing the risks associated with the widespread ability to synthesize DNA sequences.

References

[ADDS21] Martin R. Albrecht, Alex Davidson, Amit Deo, and Nigel P. Smart. Round-optimal verifiable oblivious pseudorandom functions from ideal lattices. In Juan A. Garay, editor, *Public-Key Cryptography - PKC 2021 - 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10-13, 2021, Proceedings, Part II*, volume 12711 of *Lecture Notes in Computer Science*, pages 261–289. Springer, 2021. 2

[AJL$^+$12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 483–501. Springer, 2012. 5

[FIPR05] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 303–324. Springer, 2005. 2

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. 2

[ILL89] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 12–24, New York, NY, USA, 1989. Association for Computing Machinery. 5

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007. 4

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and kdcs. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 327–346. Springer, 1999. 2

[oST20] National Institute of Standards and Technology. Guideline for using cryptographic standards in the federal government: Cryptographic mechanisms. Technical report, 2020. 2

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005. 5

[RP] Abhi Shelat Rafael Pass. A course in cryptography. 6