

Revisiting OKVS-based OPRF and PSI: Cryptanalysis and Better Construction

Kyoohyung Han¹, Seongkwang Kim¹, Byeonghak Lee¹, and Yongha Son^{2*}

¹ Samsung SDS, Seoul, Korea,
{kh89.han,sk39.kim,byghak.lee}@samsung.com
² Sungshin Women's University, Seoul, Korea,
yongha.son@sungshin.ac.kr

Abstract. Oblivious pseudorandom function (OPRF) is a two-party cryptographic protocol that allows the receiver to input x and learn $F(x)$ for some PRF F , only known to the sender. For private set intersection (PSI) applications, OPRF protocols have evolved to enhance efficiency, primarily using symmetric key cryptography. Current state-of-the-art protocols, such as those by Rindal and Schoppmann (Eurocrypt '21), leverage vector oblivious linear evaluation (VOLE) and oblivious key-value store (OKVS) constructions.

In this work, we identify a flaw in an existing security proof, and present practical attacks in the malicious model, which results in additional PRF evaluations than the previous works' claim. In particular, the attack for malicious model is related to the concept of OKVS overfitting, whose hardness is conjectured in previous works. Our attack is the first one to discuss the concrete hardness of OKVS overfitting problem.

As another flavour of contribution, we generalize OKVS-based OPRF constructions, suggesting new instantiations using a VOLE protocol with only Minicrypt assumptions. Our generalized construction shows improved performance in high-speed network environments, narrowing the efficiency gap between the OPRF constructions over Cryptomania and Minicrypt.

Keywords: oblivious pseudorandom function, oblivious key-value store, private set intersection

1 Introduction

Oblivious pseudo random function (OPRF) is a two-party cryptographic protocol that allows the receiver to input x and learn $F(x)$ for some PRF F , only known to the sender. OPRF in general can be thought of as a two-party protocol where the sender inputs a secret key and the receiver inputs some items to be evaluated and obtains a PRF value for each input. In contrast, OPRF for PSI application – which will be called *batch OPRF* – have been advanced in an independent

* This work was done while Y. Son was at Samsung SDS.

direction. As two-party PSI does not require the sender to choose the secret key nor to evaluate each item separately, batch OPRF usually outputs a random secret key to the sender and batched PRF evaluations to the receiver. Any item which is not in the batched inputs at the moment of invoking the OPRF protocol cannot be evaluated.

At the cost of such demerits, batch OPRFs are concretely efficient since they heavily use symmetric key cryptography rather than public key cryptography such as Diffie-Hellman computation. Pinkas et al. [9] proposed a PSI protocol based on a special sort of data structure called oblivious key-value store (OKVS) (called PaXoS at that time of writing). Although this paper does not include any explicit OPRF construction, the PSI protocol can be naturally extend to a PSI protocol, which we call PRTY construction hereafter. After then, Rindal and Schoppmann proposed an improved construction while replacing a subroutine of the PRTY construction by another functionality called vector-oblivious linear evaluation (VOLE), which forms the state-of-the-art protocols of (batch) OPRF and PSI protocols [10,3], and we call this by RS construction.

Currently, the RS construction that utilizes VOLE over large field such as $\text{GF}(2^{128})$ has much better performance than the PRTY construction, in both computation and communication view. It depends on so-called VOLE protocols based on pseudorandom correlation generator (PCG) [5,4,11], which assumes some (some variants of) Learning Parity with Noise (LPN). Meanwhile, the PRTY construction used the OOS functionality [8] (or $\text{GF}(2)$ -VOLE), which can be realized with only Minicrypt assumptions. However, as the performance gap between two constructions is fairly large currently, the advantage of PRTY construction in robustness may seem less attractive.

In batch OPRF protocols based on OKVS, the receiver encodes all its input items into an OKVS to obtain the PRF values of them. Because of the linearity of existing OKVS, the OPRF protocols based on OKVS by nature allows more evaluations than it should. Pinkas et al. upper bound the allowed number of evaluations information-theoretically, and Garimella et al. formalize the problem to overpack in an OKVS as *OKVS overfitting problem*.

1.1 Our Contribution

We revisit the security of batch OPRFs, with respect to the number of evaluation. Although OPRF protocols should prevent the receiver to arbitrarily evaluate the PRF value, we found that batch OPRF protocols in [13,10,3] based on OKVS allow more evaluations than the authors claimed. We point out the flaw in the security proof, and present practical attacks on them.

In the malicious model, we propose an overfitting algorithm to solve the OKVS overfitting problem, whose main idea is to reduce the overfitting problem of OKVS to either a k -XOR problem or a multicollision finding problem. To the best of our knowledge, the second attack is the first constructive (not information-theoretical) solving algorithm for the OKVS overfitting problem [7]. To prevent these attacks, [13] in the malicious setting incurs 1371% communication overhead, where the number of items is set to 2^{20} .

In PSI applications, our attacks usually incur a situation that a corrupt receiver knows “the sender does not have some specific items”. In the literature, the security of PSI only refers to the opposite side (“the sender have some specific items”). However, we clarify that such situation may leak some membership information by computing statistical distance. Depends on the application setting, the distance may exceed $2^{-\lambda}$ where λ is the statistical security parameter.

In a construction aspect, we suggest some possible mitigations to prevent our proposed attacks, and also provide revised security proof along with the mitigations in the malicious model. The mitigations shows more efficiency compared to solely following the PRTY information-theoretic bound.

We also investigate more general instantiations of OKVS-based OPRF construction, and suggest new instantiations that generalize the PRTY construction [9] using a recently proposed VOLE protocol over Minicrypt assumptions [14]. In the fast network environments, our generalization even outperforms the RS construction (with quasi-cyclic code). In slower network environment, as our generalization requires more communication than the RS constructions, the RS constructions remains the best one. Although, our generalization is still meaningful in a view that it narrows the communication gap between the RS construction (Cryptomania) and the PRTY construction (Minicrypt) from 4.2x to 1.3x.

2 Preliminaries

2.1 Notation

For a matrix A , each i -th row vector is denoted by \vec{A}_i and j -th column vector is denoted by \vec{A}^j . For a vector $\vec{\Delta} = (\Delta_1, \dots, \Delta_n)$ and a matrix U , we denote $\vec{\Delta} \odot U := (\Delta_1 \cdot \vec{U}^1, \dots, \Delta_{n_c} \cdot \vec{U}^{n_c})$. Unless otherwise stated, every field \mathbb{F} in our paper is assumed to be of characteristic 2, or we explicitly write the finite field (or Galois field) of size q by $\text{GF}(q)$. For a field \mathbb{F} , we write a linear code C in a k_c dimensional subspace in \mathbb{F}^{n_c} with minimum distance d_c by $[n_c, k_c, d_c]_{\mathbb{F}}$, with an exception of $[n_c, k_c, d_c]_2$ for $\mathbb{F} = \text{GF}(2)$ case. We often consider a $[n_c, k_c, d_c]_{\mathbb{F}}$ linear code as a map $C : \mathbb{F}^{k_c} \rightarrow \mathbb{F}^{n_c}$ to write the codeword on $\vec{x} \in \mathbb{B}^{k_c}$ by $C(\vec{x})$. We denote computational security parameter by κ , and statistical security parameter by λ . We denote $B(n, p)$ binomial distribution of probability p and n independent experiments.

2.2 Vector Oblivious Linear Evaluation

A (subfield) vector oblivious linear evaluation, precisely (\mathbb{F}, \mathbb{B}) -VOLE outputs two parties a secret shares of scalar-vector multiplication $\Delta \cdot \vec{U}$ for randomly chosen $\Delta \in \mathbb{F}$ and $\vec{U} \in \mathbb{B}^m$ to each party. When the subfield \mathbb{B} equals to \mathbb{F} , we call it simply by \mathbb{F} -VOLE. The corresponding ideal functionality $\mathcal{F}_{\text{vole}}$ is defined as Figure 1.

During the past few years, the performance of VOLE for large fields such as $\text{GF}(2^{128})$ has been rapidly improved, thanks to the advances on pseudorandom

correlation generator (PCG) [5] based on learning with parity (LPN) problem. However, such rapid performance is enabled by assuming the hardness of LPN over somewhat non-standard codes [6,4,11]. Indeed, there has been proposed an attack [11] on the silver code utilized in [6]. Meanwhile, Roy [14] proposed a novel VOLE protocol that only requires Minicrypt assumptions, unlike the LPN-based protocols. It is practically efficient for small field such as $\text{GF}(2^f)$ with $f \leq 8$.

<p>Parameters: Two parties (a sender and a receiver). A field \mathbb{F} with a subfield \mathbb{B}, and an integer m representing the length of output vector.</p> <p>Functionality:</p> <ul style="list-style-type: none"> – If the receiver is malicious, wait for them to send $\vec{V} \in \mathbb{F}^m$ and $\vec{U} \in \mathbb{B}^m$. Then sample $\Delta \leftarrow \mathbb{F}$ and let $\vec{W} := \vec{V} + \Delta \cdot \vec{U} \in \mathbb{F}^m$. – If the sender is malicious, wait for them to send $\vec{W} \in \mathbb{F}^m$ and $\Delta \in \mathbb{F}$. Then sample $\vec{U} \leftarrow \mathbb{B}^m$ and let $\vec{V} := \vec{W} - \Delta \cdot \vec{U} \in \mathbb{F}^m$. – Otherwise, i.e., adversarial party is semi-honest, sample $\vec{V} \leftarrow \mathbb{F}^m, \vec{U} \leftarrow \mathbb{B}^m$ and $\Delta \leftarrow \mathbb{F}$, and let $\vec{W} = \vec{V} + \Delta \cdot \vec{U} \in \mathbb{F}^m$ <p>Output $\vec{W} \in \mathbb{F}^m$ and $\Delta \in \mathbb{F}$ to the sender, and $\vec{V} \in \mathbb{F}^m$ and $\vec{U} \in \mathbb{B}^m$ to the receiver.</p>

Fig. 1: An ideal functionality of $\mathcal{F}_{\text{vole}}(\mathbb{F}, \mathbb{B})$ for (\mathbb{F}, \mathbb{B}) -vector oblivious linear evaluation

2.3 Oblivious Key-Value Store

Informally, an oblivious key-value store (OKVS) is a data structure that efficiently encodes n pairs of keys and values, which satisfies if a value are random, the corresponding key cannot be recovered from the encoding of the key-value pairs.

Definition 1 (Oblivious Key-Value Store). *An oblivious key-value store (OKVS) with key universe \mathcal{K} and value universe \mathcal{V} consists of two functions:*

- $\text{Ecd} : (\mathcal{K} \times \mathcal{V})^n \rightarrow \mathcal{V}^m \cup \{\perp\}$, a function that receives n distinct key-value pairs $(k_i, v_i)_{i \in [n]}$ then outputs encoding S or failure symbol \perp ;
- $\text{Dcd} : \mathcal{V}^m \times \mathcal{K} \rightarrow \mathcal{V}$, a function that receives encoding S and a key k then outputs the associated value v .

For correctness, for all $I \subset \mathcal{K} \times \mathcal{V}$ of n elements with distinct keys and an ordering $\vec{I} \in (\mathcal{K} \times \mathcal{V})^n$ of I , an OKVS should satisfies

$$(k, v) \in I \text{ and } \text{Ecd}(\vec{I}) = S \neq \perp \Rightarrow \text{Dcd}(S, k) = v.$$

For obliviousness, for any pair of lists of n distinct keys $(k_1, \dots, k_n) \in \mathcal{K}^n$ and $(k'_1, \dots, k'_n) \in \mathcal{K}^n$ and n random values $v_1, \dots, v_n \leftarrow_{\S} \mathcal{V}$, $\text{Ecd}((k_1, v_1), \dots, (k_n, v_n))$ and $\text{Ecd}((k'_1, v_1), \dots, (k'_n, v_n))$ should be computationally indistinguishable.

The storage efficiency of OKVS can be measured by the expansion ratio of the number of key-value pairs n and the length of the encoding vector m . Many known OKVS constructions achieve $m = (1 + \varepsilon_{\text{okvs}}) \cdot n$ for some small constant $\varepsilon_{\text{okvs}}$: PaXoS [9] achieves $\varepsilon \approx 1.4$, 3H-GCT [7] and RR22 [10] achieve $\varepsilon \approx 0.3$, and RB-OKVS [3] achieve ε down to 0.03.

Several OKVS applications, such as OPRF and PSI, expect OKVS to have linearity and support for some sort of homomorphic operations, and we call such OKVS by *linear OKVS*.

Definition 2 (Linear OKVS). An OKVS is linear if there exists a function $\text{row} : \mathcal{K} \rightarrow \mathcal{V}^m$ such that $\text{Dcd}(S, k) = \langle \text{row}(k), S \rangle$ for all $k \in \mathcal{K}$ and $S \in \mathcal{V}^m$. If the range of such function row can be restricted to a set of binary vectors, i.e., $\text{row} : \mathcal{K} \rightarrow \{0, 1\}^m$, we call the OKVS as binary linear, or simply binary.

3 Oblivious PRF and OKVS-based Constructions

This section provides a formal definition of oblivious PRF (and private set intersection), and reviews the OKVS-based OPRF constructions [9,13], which consists of the state-of-the-art protocols [10,3].

3.1 Ideal Functionalities and Generic PSI Construction

The ideal functionality of an oblivious pseudorandom function (OPRF) is described in Figure 2, and the (two-party) private set intersection (PSI) functionality is described in Figure 3. Ideally, the receiver should not be able to obtain any OPRF evaluations other than for its input. However, the OPRF definition in Figure 2 allows more PRF evaluations, denoted by $n' > n$ for a malicious receiver, and the PSI definition also allows at most n' items for the malicious receiver. This reflects the fact that known OPRF (or PSI) protocols enable an

Parameters: Two parties (a sender and a receiver). The receiver’s set size parameters n for the semi-honest model and n' for the malicious model. An OPRF output length ℓ_2 .

Functionality: Upon the receiver’s input set X , abort if $|X| > n'$ when the receiver is malicious.

The functionality defines a random function $F : \{0, 1\}^* \rightarrow \{0, 1\}^{\ell_2}$, and output $F(X) = \{F(x) \mid x \in X\}$ to the receiver. After then, upon the sender’s query y , the functionality sends $F(y)$ to the Sender.

Fig. 2: Ideal functionality $\mathcal{F}_{\text{oprf}}$ of oblivious pseudorandom function.

Parameters: Two parties (a sender and a receiver). The sender’s set size parameter n_y , and the receiver’s set size parameters n_x for the semi-honest model and n' for the malicious model.

Functionality: Upon the receiver’s input set X and the Sender’s input set Y , abort if $|X| > n'$ when the receiver is malicious, and outputs $X \cap Y$ to the receiver.

Fig. 3: Ideal functionality \mathcal{F}_{psi} of (2-party) private set intersection.

adversary to learn OPRF values $F(X)$ (or $X \cap Y$) for some $n' (> n)$ -sized set X , while pretending to run the protocol with a size n set. This type of definition has been widely adopted in the literature [9,13].

PSI from OPRF. Given an OPRF functionality $\mathcal{F}_{\text{oprf}}$, it is straightforward to construct a protocol that realizes \mathcal{F}_{psi} , as shown in Figure 4. This protocol requires one additional round of communication, with $\ell_2 \cdot n_y$ bits transferred from the sender to the receiver, on top of the communication cost for realizing $\mathcal{F}_{\text{oprf}}$. The concrete choice of ℓ_2 has been improved using better security proofs, and our paper adapts the state-of-the-art result from [13].

Parameters: Two parties (a sender and a receiver). An OPRF functionality $\mathcal{F}_{\text{oprf}}$ with set size parameters n_x or n' , and OPRF length ℓ_2 .

Protocol: Upon an input set X from the receiver, and an input set Y from the sender, the protocol runs as follows:

1. The sender and the receiver interact with $\mathcal{F}_{\text{oprf}}$ with the receiver’s input set X .
2. As $\mathcal{F}_{\text{oprf}}$ outputs, the receiver obtains $F(X)$. Then the sender obtains $F(Y) = \{F(y) \in \{0, 1\}^{\ell_2} : y \in Y\}$ by querying each $y \in Y$ to $\mathcal{F}_{\text{oprf}}$.
3. The sender sends $F(Y)$ to the receiver in a random order, who outputs $Z = \{x \in X : F(x) \in F(Y)\}$.

Fig. 4: Protocol Π_{psi} for a private set intersection using $\mathcal{F}_{\text{oprf}}$.

Theorem 1 (Adapted from [13]). *The protocol Π_{psi} realizes the \mathcal{F}_{psi} functionality in the semi-honest model with $\ell_2 = \lambda + \log(n_x n_y)$ and in the malicious model with $\ell_2 = \kappa$, in a $\mathcal{F}_{\text{oprf}}$ -hybrid model.*

Remark 1. In more detail, the simulated view against the malicious sender in [13] is distinguishable from the real protocol execution with $2^{\ell_2}/n_x$ random oracle queries. Hence, the simulation with $\ell_2 = \kappa$ can be distinguished by fewer than

2^κ queries. Rindal and Schoppmann was already aware of this fact while they stucked to use $\ell_2 = \kappa$ [13]. To be more rigorous, it is required to set $\ell_2 = \kappa + \log n_x$ in order to simulate against the malicious sender. But we follow the choice $\ell_2 = \kappa$ as the distinguishing advantage does not lead to any actual information leakage.

3.2 OKVS-based OPRF Constructions

We present an overview of the OKVS-based OPRF construction in Figure 5, which captures the RS construction [13,10,3] with $\mathbb{F} = \text{GF}(2^{128})$ and the identity linear code $[1, 1, 1]_{\text{GF}(2^{128})}$, as well as the natural OPRF extension from the PRTY PSI protocol [9,7] with $\mathbb{F} = \text{GF}(2)$ and non-trivial binary linear codes. Detailed parameter selections and security arguments will be presented in later sections, as one of our main contributions is to identify the flaws in previous works.

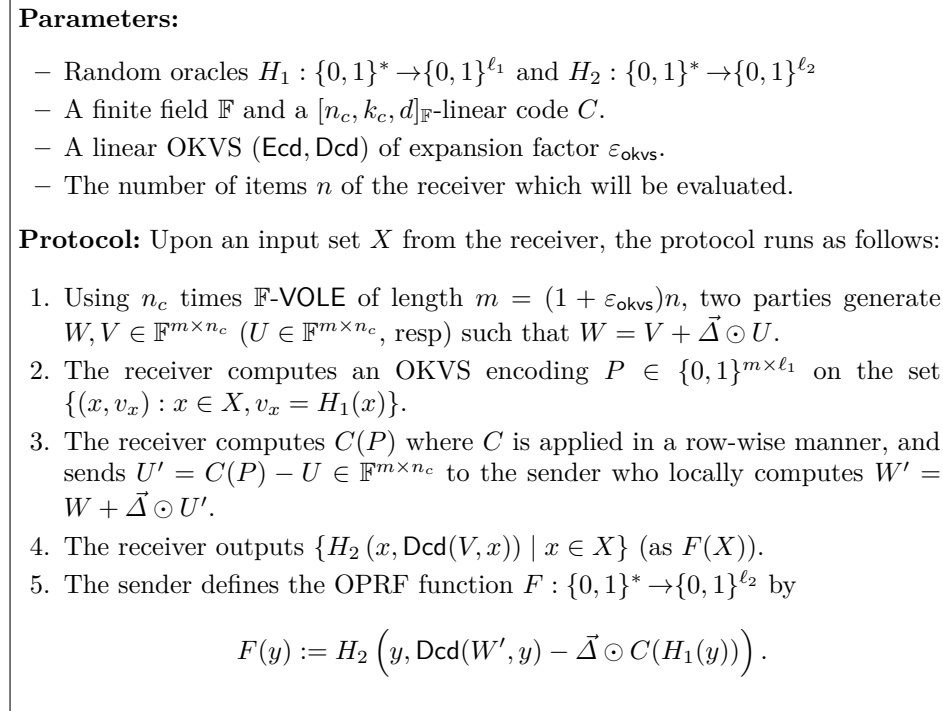


Fig. 5: Integrated overview of OKVS-based OPRF protocols.

To see the correctness, the set $\{H_2(x, \text{Dcd}(V, x)) \mid x \in X\}$ should equal to $F(X)$ with the sender's definition $F(y) = H_2(y, \text{Dcd}(W, y) - \vec{\Delta} \odot C(H_1(y)))$.

From the linearity of Dcd and the linear code C , we have

$$\begin{aligned} F(y) &= H_2(y, \text{Dcd}(W', y) - \vec{\Delta} \odot C(H_1(y))) \\ &= H_2(y, \text{Dcd}(V, y) - \vec{\Delta} \odot C(\text{Dcd}(P, y) - H_1(y))) \end{aligned}$$

for any $y \in \{0, 1\}^*$. So, the OKVS correctness is ensured by the fact that $\text{Dcd}(P, x) = H_1(x)$ for every $x \in X$. Thus, we further have

$$F(x) = H_2(x, \text{Dcd}(V, x) - \vec{\Delta} \odot C(\text{Dcd}(P, x) - H_1(x))) = H_2(x, \text{Dcd}(V, x)). \quad (1)$$

Remark 2. The original description in [9] used OOS (correlated) OT extension [8] instead of VOLE over $\text{GF}(2)$, but two functionalities are exactly same.

Overfitted OKVS. The key to the correctness of OKVS-based OPRF in (1) is the equality $\text{Dcd}(P, x) = H_1(x)$ for every $x \in X$, ensured by the OKVS correctness. However, in OKVS-based OPRF protocols, a malicious receiver can arbitrarily generate the OKVS encoding P , allowing more than n items x such that $\text{Dcd}(P, x) = H_1(x)$. This enables the receiver to obtain the PRF values for these x . Garimella et al. [7] formalize this issue as the *OKVS overfitting game* as follows.

Definition 3 ((n, n') -OKVS overfitting game, [7]). *Let (Ecd, Dcd) be an OKVS with parameters chosen to support n items, and let $H_1 : \{0, 1\}^* \rightarrow \{0, 1\}^{\ell_1}$ be a random oracle. For any arbitrary PPT adversary \mathcal{A} that outputs $P \in \{0, 1\}^{\ell_1 \times m} \leftarrow \mathcal{A}^H(1^\kappa)$, define*

$$X' = \{x \mid \mathcal{A} \text{ queried } H_1 \text{ at } x \text{ and } \text{Dcd}(P, x) = H_1(x)\}.$$

If $|X'| > n'$, then the adversary wins the (n, n') -OKVS overfitting game.

We say the (n, n') -OKVS overfitting problem is hard for an OKVS construction if no PPT adversary wins this game except with negligible probability.

PRTY bound. It can be easily observed that if the underlying OKVS is linear, a malicious receiver can obtain $m = (1 + \varepsilon_{\text{okvs}}) \cdot n$ PRF evaluations with almost no computational overhead. This fact leads OKVS-based OPRF protocols to focus on $n' = c \cdot m$ for some $c > 1$.

In [9], Pinkas et al. analyzed the choice of ℓ_1 (the output bit-length of H_1 in Figure 5) that makes the OKVS-based OPRF construction information-theoretically secure against malicious adversaries. This analysis can be rephrased using OKVS terminology as follows.

Lemma 1 (PRTY bound, [9]). *Suppose an adversary makes q queries to random oracle H_1 with output length ℓ_1 , and then generates an OKVS P of size m . For a fixed integer n' , let \mathcal{E} denote the event that $\text{Dcd}(P, x) = H_1(x)$ for at least n' values x that were queried to H_1 . Then,*

$$\Pr[\mathcal{E}] \leq \frac{\binom{q}{n'}}{2^{(n'-m)\ell_1}}.$$

Suppose there is a linear system $Ax = b$ where $A \in (\mathbb{F}_{2^{\ell_1}})^{n' \times m}$ ($n' > m$) is invertible and $b \leftarrow_{\S} (\mathbb{F}_{2^{\ell_1}})^m$. The probability that b produces a solvable system is $2^{(n'-m)\ell_1}$. Since the number of n' -tuples of queried items is $\binom{q}{n'}$, the bound in Lemma 1 is quite tight, with only a small gap possible due to a non-invertible OKVS system.

4 Security Flaws of OKVS-based OPRFs

The OKVS-based OPRF naturally allows the receiver to locally compute $F(x)$ that satisfies $\text{Dcd}(P, x) = H_1(x) \in \{0, 1\}^{\ell_1}$. Thus, the length ℓ_1 should be set properly to prevent unwanted PRF evaluations. Indeed, previous OKVS-based OPRF protocols suggested some appropriate choice of ℓ_1 to bound the number of PRF evaluation by n (semi-honest) or n' (malicious), along with corresponding security arguments. However, in this section, we present some (possible) vulnerabilities in the previous works setting on n, n' and ℓ_1 , and show that it indeed implies more number of PRF evaluations than the works claimed.

4.1 Caution for Possible Semi-honest OPRF

Although previous works [9,13,10] claimed the malicious security of the OPRF protocol or the semi-honest security of the PSI protocol rather than the semi-honest security of OPRF protocol, one can naturally derive a semi-honest version of OPRF protocol and try to use it. In this section, we briefly point out which security issue can be popped up.

All semi-honest OKVS-based PSI protocols set $\ell_1 = \lambda + 2 \log n$ where n is the number of items in both parties, whose rationale is to prevent unwanted collision of hashed values. However, as random oracle is an idealization of cryptographic hash function, it should be assumed that the receiver is free to query to the random oracle before or after the protocol. It implies that the probability of the equality $\text{Dcd}(P, x) = H_1(x)$ for a fixed P and a random x is $1/2^{\ell_1}$.

This fact discloses the trivial attack on the natural OPRF extension; for a fixed P after the OPRF protocol, the corrupt receiver can query to H_1 and find inputs satisfying $\text{Dcd}(P, x) = H_1(x)$. As the computational security parameter κ is much larger than those choices of ℓ_1 , the receiver can obtain q/ℓ_1 extra evaluations with q local computation of H_1 .

4.2 Malicious Flaw: Hardness of OKVS Overfitting Game

The OPRF (PSI) protocol of PRTY [9] takes ℓ_1 so that $\Pr[\mathcal{E}]$ in the PRTY bound (Lemma 1) is bounded by $2^{-\lambda}$, given $n' = c \cdot m$. One can check that larger n' yields smaller ℓ_1 , and such smaller ℓ_1 naturally brings better efficiency. Indeed, [9] set $n' \approx 12n$ to have efficient protocol where they reported experimental results. Meanwhile, Rindal and Schoppmann [13] (implicitly) argued that n' can be limited by only $m = (1 + \varepsilon_{\text{okvs}})n$, by setting $\ell_1 = \kappa$. This obviously makes it possible to take smaller ℓ_1 than [9] protocol, so that has better parameter choice

and better performance of the OPRF protocol.³ This has also been continuously adapted in the following works [10,3]; whose main points are improving OKVS performance while following the framework of [13].

However, we claim the choice of $\ell_1 = \kappa$ of RS construction with $n' = m$ is flawed. The reason is simple: $\ell_1 = \kappa$ is less than ℓ_1 derived from the PRTY bound with $n' = m$. That is, the malicious receiver with unbounded power can indeed find (more than) n' items x fitting in some P . However, such presence of unbounded adversary could be not considered as serious one, because one can simply assume the *computational* hardness of OKVS overfitting problem to make [13] secure. Indeed, Garimella et. al. [7] stated that OKVS overfitting could be *computationally* hard even ℓ_1 is below PRTY bound.

What we further make is the concrete attack that shows OKVS overfitting can be computationally done, so that $\ell_1 = \kappa$ choice of [13,10,3] allows more PRF evaluations than $n' = m$ using less than 2^κ computational cost. The details are placed in the next section independently.

Flaw in the security proof. The Hybrid 4 of Lemma 2 in [13] is the relevant argument that argues n' can be limited to $m = (1 + \varepsilon_{\text{okvs}})n$. However, their argument only considered the case where the malicious adversary fixes the OKVS encoding P and then expects that $\text{Dcd}(P, x) = H_1(x)$ for additional x . This makes the proof false, because the malicious adversary can make $q = O(2^\kappa)$ amount of H_1 queries in advance, and then tries to find some maximal subset X' that overfits to some P of size m among the H_1 queries.

4.3 Efficacy of Extra Evaluation

In the OPRF view, extra evaluation directly violates the security requirements. However, the implication on PSI need some consideration. If the receiver obtains an additional PRF evaluation $F(y)$ for $y \in Y \setminus X$, this immediately violates the PSI functionality since the receiver knows other information than the intersection $X \cap Y$. The case where the additional PRF evaluation $F(y)$ is for $y \notin X \cup Y$ lets the receiver know that the element y is *not in* the sender's set Y , whose effect seems a bit ambiguous.

In fact, we can measure the amount of information leakage from the information $y \notin \bar{Y}$. Suppose that \mathcal{D}_1 be the original distribution of the sender's dataset \bar{Y} with $p_1(\bar{Y}) = \Pr_{Y \sim \mathcal{D}_1}[Y = \bar{Y}]$. And, let \mathcal{D}_2 be the distribution of the dataset without an element z in a universe of items in set. From the additional leakage of $z \notin \bar{Y}$, the corrupt receiver now have the distribution \mathcal{D}_2 of the sender's dataset. And the probability function of \mathcal{D}_2 can be computed as follows:

$$p_2(\bar{X}) = \begin{cases} \frac{1}{1-p'} \cdot p_1(\bar{X}) & \text{if } z \notin \bar{X} \\ 0 & \text{otherwise} \end{cases}$$

³ We are not saying that every performance gain of [13] comes from such small ℓ_1 ; indeed the primary change from [9] to [13] that greatly affects to the performance is replacing OOS functionality into LPN-based VOLE.

where $p' = \sum_{\{\bar{X}|z \in \bar{X}\}} p_1(\bar{X})$. Note that the probability $p_2(\bar{X})$ is increased proportionally to have sum of all the probability 1 when $z \notin \bar{X}$. Now we can compute the statistical distance between \mathcal{D}_1 and \mathcal{D}_2 ,

$$\text{dist}(\mathcal{D}_1, \mathcal{D}_2) = \sum_{z \notin \bar{X}} |p_1(\bar{X}) - p_2(\bar{X})| + \sum_{z \in \bar{X}} |p_1(\bar{X}) - p_2(\bar{X})| = 2p'.$$

From this, we conclude that the information of $y \notin Y$ can lead to a non-negligible amount of information, if y has a non-negligible probability ($> 2^{-\lambda}$) to be included in the honest party's dataset. Moreover, this series of computations shows an undesirable feature of the PSI protocol, which is the dependency of the security on the distribution of the dataset.

5 Overfitting Attacks on Various OKVS

The ℓ_1 bound in [13] was found out to be information-theoretically not secure. To connect the information-theoretic vulnerability to a concrete attack, we need to overfit the OKVS P . Overfitting P refers to a problem to find $n' > m$ items $x_1, \dots, x_{n'}$ and a vector P that satisfy $\text{Dcd}(P, x_i) = H_1(x_i)$ for all i . Since the OKVS algorithm used in [13] is binary linear, there is a function $\text{row} : \{0, 1\}^* \rightarrow \{0, 1\}^{\ell_1}$ such that $\text{Dcd}(P, x) = \langle \text{row}(x), P \rangle$. Then overfitting P is in principle a finding $\vec{X} = (x_1, \dots, x_{n'})$ which builds a redundant (i.e., linearly dependent) system of linear equations $\text{row}(\vec{X}) \cdot P = H_1(\vec{X})$ for $n' > n$, where $\text{row}(\vec{X})$ (and $H_1(\vec{X})$) is a matrix whose i -th row is $\text{row}(x_i)$ (and $H_1(x_i)$).

For the idea of the attacks, observe that if there are some elements x_1, \dots, x_k in \vec{X} satisfy that $\sum_{j=1}^k \text{row}(x_j) \parallel H_1(x_j) = 0$, they contribute to the rank of $\text{row}(\vec{X})$ by at most $k - 1$. Finding such x_1, \dots, x_k is equivalent to solving the k -XOR problem for $k \geq 2$, and hence we can build \vec{X} of length $k(m-1)/(k-1) > m$ by solving the k -XOR problem repeatedly. Although $\text{row}(\vec{X})$ is a quite large matrix of size $k \times m$ where k -XOR problem is difficult in general, the attacks can be practically feasible thanks to the special structure of $\text{row}(\vec{X})$ observed in previous OKVS constructions [13,10,3], whose details are described in the following subsections.

Before presenting the attacks, we provide some definitions of basic problems required to formulate the attacks.

k -XOR Problem and Multi-collision Finding Problem. Before explaining our attacks, we briefly introduce the k -XOR problem and the multi-collision finding problem since it is a crucial subroutine of our attacks to solve these problems. The k -XOR problem is to find a k -tuple in some lists whose sum is 0. Formally, a k -XOR problem is defined as follows.

Definition 4 (k -XOR Problem). *Given lists of n -bit strings L_1, \dots, L_k , find k distinct elements $a_1 \in L_1, \dots, a_k \in L_k$ such that their XOR sum is zero:*

$$a_1 \oplus a_2 \oplus \dots \oplus a_k = 0.$$

The state of the art algorithm to solve the k -XOR problem is Wagner's k -tree algorithm [17]. It costs $O(k \cdot 2^{n/(1+\lceil \log k \rceil)})$ time and space with lists of size $O(2^{n/(1+\lceil \log k \rceil)})$ each.

For a given cryptographic hash function, the multi-collision finding problem is to find c multi-collisions which give the same hash output. Formally, a multi-collision finding problem is defined as follows.

Definition 5 (c -Multicollision Finding Problem). *Given a hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^n$, find c distinct elements a_1, \dots, a_c such that:*

$$H(a_1) = H(a_2) = \dots = H(a_c).$$

The best algorithm to solve the c -multicollision finding problem is proposed by Suzuki et al. [15], which costs $O((c!)^{1/c} \cdot 2^{(c-1)n/c})$ time and space.

5.1 Overfitting PaXoS

PaXoS [9] is the firstly proposed OKVS with linear complexity. Given a vector of input items \vec{X} , which is an ordering of the set X , PaXoS has a following type of the $\text{row}(\vec{X})$ matrix.

$$\text{row}(\vec{X}) = [R_{\vec{X}} \mid D_{\vec{X}}]$$

where $R_{\vec{X}} \in \mathbb{F}_2^{n \times m}$ is defined by a pair of hash functions (h_1, h_2) each to $[m] = \{1, \dots, m\}$, and $D_{\vec{X}} \in \mathbb{F}_2^{n \times d}$ is a matrix with uniformly random entries. Each row of $R_{\vec{X}}$ corresponds to an item x , and it has only two 1's whose positions are $h_1(x)$ and $h_2(x)$. In the same row, the row of $D_{\vec{X}}$ also corresponds to the same item x , and it is defined by a uniform hash function $h_0 : \{0, 1\}^* \rightarrow \{0, 1\}^d$. We note that m is set to be $2.4n$ in [13].

As $R_{\vec{X}}$ is a sparse matrix, it is hard to apply Wagner's k -tree algorithm directly on $\text{row}(x) \parallel H_1(x)$. So, we will solve the k -XOR problem over $h_0(x) \parallel H_1(x)$ rather than $\text{row}(\vec{X}) \parallel H_1(x)$. To make $R_{\vec{X}}$ part also sum to zero, we bucketize $\text{row}(\vec{X}) \parallel H_1(x)$ by h_1 and h_2 , and deliberately organize a singular combination of buckets. For an easy example, if three items x_1, x_2, x_3 satisfy

$$\begin{aligned} h_1(x_1) &= 1, h_2(x_1) = 2, \\ h_1(x_2) &= 2, h_2(x_2) = 3, \\ h_1(x_3) &= 3, h_2(x_3) = 1, \end{aligned}$$

then $R_{\vec{X}}$ part of those three items sum to zero. Then, solving 3-XOR problem in those three buckets gives a small redundant system of equations of rank 2. Our attack on [13] basically repeats these steps until $\text{row}(\vec{X})$ has full rank, but the number of buckets may vary. In the following, we describe the detailed procedure of the attack.

- (Case $n' < 2(m-1)$) For this case, we reduce the $(n, \frac{k(m-1)}{k-1})$ -overfitting game to a k -XOR problem as follows. We will assume that $k-1 \mid m-1$, but the attack also works well otherwise.

1. Let $Q = \{x_1, \dots, x_q\}$ be an arbitrary subset in $\{0, 1\}^*$. Query all the items in Q to h_0, h_1, h_2 , and H_1 . Bucketize Q by $\{h_1, h_2\}$, denoting B_{h_1, h_2} the corresponding bucket. Then, there are $\binom{m}{2}$ buckets, and there are expectedly $q/\binom{m}{2}$ items per bucket.
2. Make a $k \times k$ -matrix K over \mathbb{F}_2 such that
 - $\text{rank}(K) = k - 1$;
 - $\text{rank}(K') = k - 1$ where $K' \in \mathbb{F}_2^{(k-1) \times (k-1)}$ is sub-matrix of K with first $k - 1$ rows and first $k - 1$ columns;
 - each row of K should have only two 1's.
 See Figure 6 for examples.
3. Set $X' \leftarrow \emptyset$, and do the following for $j \in \{1, 1 + (k - 1), 1 + 2(k - 1), \dots, m - k + 2\}$.
 - (a) Denote the positions of two 1's in the i -th row ($1 \leq i \leq k$) of K by p_i and p'_i . Solve a k -XOR problem for $h_0(x) \parallel H_1(x)$ in buckets $B_{j+p_1, j+p'_1}, \dots, B_{j+p_k, j+p'_k}$.
 - (b) Denote the solution (x_1, \dots, x_k) , then it satisfies

$$\sum_{i=1}^k h_0(x_i) \parallel H_1(x_i) = 0.$$

Set $X' \leftarrow X' \cup \{x_1, \dots, x_k\}$

4. Let $\vec{X}' = (x'_1, \dots, x'_{n'})$ be an ordering of X' where $n' = \frac{k(m-1)}{k-1}$. Since $\text{rank}(\text{row}(\vec{X}')) = \text{rank}(\text{row}(\vec{X}') \parallel H_1(\vec{X}'))$, there is a solution P' of the linear equation $\text{row}(\vec{X}') \cdot P' = H_1(\vec{X}')$ where $H_1(\vec{X}') = (H_1(x'_1), \dots, H_1(x'_{n'}))$.

As solving a k -XOR problem of $(d + \ell_1)$ -bit strings costs $O\left(k 2^{\frac{d+\ell_1}{1+\lceil \log k \rceil}}\right)$ time, our attack costs $O\left(m^2 2^{\frac{d+\ell_1}{1+\lceil \log k \rceil}}\right)$ time including time for querying to the random oracle. Note that the time cost for random oracle query dominates the cost for solving k -XOR problems.

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

(a) $k = 4$

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

(b) $k = 5$

Fig. 6: Examples of Step 2.

- (Case $n' \geq 2m$) For this case, we reduce the $(n, c(m-1))$ -overfitting game to a c -multicollision ($c \geq 2$) finding problem as follows.
 1. Let $Q = \{x_1, \dots, x_q\}$ be an arbitrary subset in $\{0, 1\}^*$. Query all the items in Q to h_0, h_1, h_2 , and H_1 . Bucketize Q by $\{h_1, h_2\}$, denoting B_{h_1, h_2} the corresponding bucket. Then, there are $\binom{m}{2}$ buckets, and there are expectedly $q/\binom{m}{2}$ items per bucket.
 2. Set $X' \leftarrow \emptyset$ and do the following for $j \in \{1, 2, \dots, m-1\}$.
 - (a) Find a c -multicollision for $h_0(x) \parallel H_1(x)$ from bucket $B_{j, j+1}$.
 - (b) Gather the solutions of the problem to X' .
 3. Let $\vec{X}' = (x'_1, \dots, x'_{n'})$ be an ordering of X' where $n' = c(m-1)$. Since $\text{rank}(\text{row}(\vec{X}')) = \text{rank}(\text{row}(\vec{X}')|H_1(\vec{X}'))$, there is a solution P' of the linear equation $\text{row}(\vec{X}') \cdot P' = H_1(\vec{X}')$.

As finding c -multicollision of $(d + \ell_1)$ -bit strings costs $O((c!)^{1/c} 2^{\frac{(c-1)(d+\ell_1)}{c}})$ time, our attack costs $O(m^2 2^{\frac{(c-1)(d+\ell_1)}{c}})$ time for small enough c .

5.2 Overfitting 3H-GCT or RR22

Garimella et.al. [7] proposed a 3H-GCT OKVS that generalizes PaXoS by using 3 hashes instead of 2 hashes. Raghuraman and Rindal [10] proposed a similar OKVS that has only w 1's in $R_{\vec{X}}$ part, which is almost the same with 3H-GCT for $w = 3$. Precisely, it also follows the form $\text{row}(\vec{X}) = [R_{\vec{X}}|D_{\vec{X}}]$, where $R_{\vec{X}} \in \mathbb{F}_2^{n \times m}$ is defined by a triple of hash functions (h_1, h_2, h_3) each to $[m]$, and $D_{\vec{X}} \in \mathbb{F}_2^{n \times d}$ is a matrix with uniformly random entries; 3H-GCT proposal sets $\ell_1 = 1$, and RR22 sets $\ell_1 = \kappa$. We denote $d = d'\ell_1$. Each row of $R_{\vec{X}}$ corresponds to an item x , and it has only three 1's whose positions are $h_1(x)$, $h_2(x)$ and $h_3(x)$. Each row of $D_{\vec{X}}$ also corresponds to the same item x , and it is defined by a uniform hash function $h_0 : \{0, 1\}^* \rightarrow \{0, 1\}^d$. We note that m is set to be approximately $1.3n$ in [13].

As PaXoS and 3H-GCT are similar, the attacks for PaXoS work well on this case except some details. The buckets should be labeled with three hash values $\{h_1, h_2, h_3\}$. In Step 2, since 3H-GCT uses three hash functions, the submatrix should be of the different form. Step 2 can be rephrased to attack 3H-GCT as follows.

2. For an even integer $k > 2$, make a $k \times (3k/2)$ -matrix over \mathbb{F}_2 of rank $k-1$. Each row of this matrix should have only three 1's. See Figure 7 for examples.

The asymptotic time complexity should be $O(m)$ times larger than that for PaXoS.

5.3 Overfitting RB-OKVS

RB-OKVS is an OKVS proposed by Bienstock et al. [3], whose $\text{row}(\vec{X})$ has no sparse part unlike PaXoS or 3H-GCT. Given a pair of hash functions $h_1 :$

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

(a) $k = 4$

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

(b) $k = 6$

Fig. 7: Examples of submatrices when attacking 3H-GCT.

$\{0, 1\}^* \rightarrow [m - d]$ and $h_0 : \{0, 1\}^* \rightarrow \{0, 1\}^d$, $\text{row}(x)$ for an item x is defined by

$$\text{row}(x)[i] = \begin{cases} h_0(x)[i - h_1(x)] & \text{if } h_1(x) < i \leq h_1(x) + d \\ 0 & \text{otherwise} \end{cases}$$

where $v[n]$ for a vector v denotes the n -th component of v .

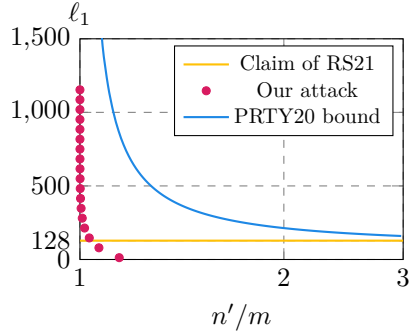
The attack can be done in the almost similar way to PaXoS and 3H-GCT. The details can be found in Appendix A due to the space limit.

5.4 Efficacy of the Attacks

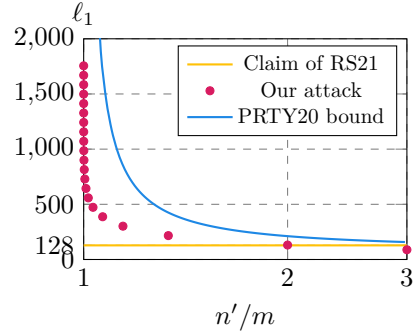
To measure the effectiveness of the attacks, we plot three dot graphs of ℓ_1 corresponding to the extra-evaluation ratio n'/m for each OKVS in Figure 8b, 8a, and 8c. Common to all the three graphs, the blue curve is the PRTY bound [9] for each parameter set, and the yellow line is ℓ_1 which was claimed secure in [13]. We note that [10,3] follow the choice of ℓ_1 . The red dots are the least ℓ_1 to prevent our attacks. Each red dot corresponds to either $3 \leq k \leq 20$ of the k -XOR problem or $2 \leq c \leq 3$ of the c -multicollision finding problem. The hash length ℓ_1 to prevent our attacks are much larger than ℓ_1 claimed in [13]. Each graph is plotted for $n = 2^{20}$, and the detailed choice of parameters is written in each figures.

When the attacks are applied in the context of PSI, it implies that a corrupt receiver can input n' random items rather than n chosen items. As data is not distributed uniformly at random in practice, n' random items seem less valuable than n chosen items so the attacks are quite useless. However, the attacks can still evaluate more items than claimed including n chosen items as follows.

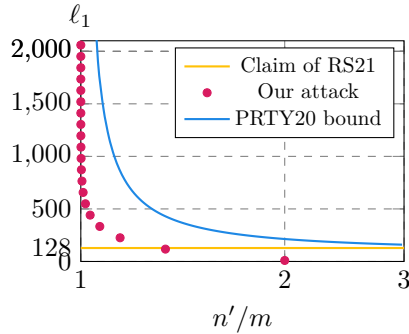
1. For a fixed (ordered) set $\vec{X} = (x_1, \dots, x_n)$, make the $\text{row}(\vec{X})$ matrix.
2. Using Gaussian elimination, find all the non-pivot positions. There are at least $(m - n)$ non-pivot positions.
3. Let Q be a set of items whose h_i ($1 \leq i \leq 3$) values are all in the non-pivot positions.
4. Mount the attack with Q as if there is no pivot positions.



(a) (RR22 [10]) We set the number of hashes 3, the expansion ratio 1.23, and the number of $\mathbb{F}_{2^{128}}$ -dense columns 2.



(b) (PaXoS [9]). We set the number of hashes 2, the expansion ratio 2.4, and the number of dense columns 40.



(c) (RB-OKVS [3]). We set the expansion ratio 1.1, and the number of dense columns 206.

Fig. 8: ℓ_1 to prevent the attack to various protocols.

It is straightforward that this variation for RS21 and RR22 can evaluate $n + \frac{n'}{m} \cdot (m - n)$ with similar complexities. For BPSY23, it is not always true. If a consecutive d positions do not include $k - 1$ non-pivot positions, the attack cannot utilize all the non-pivot positions so that the number of evaluation may be less than $\frac{n'}{m} \cdot (m - n)$. For the worst case (most of the consecutive d positions do not include $k - 1$ non-pivot positions), this variation may not evaluate more than m items.

6 Generic Security Considerations

In this section, we propose generic mitigations for our attacks, and a modified OKVS-based OPRF with provably secure choice of parameters.

6.1 Mitigations for Attacks and Revised Parameter Selection

For the semi-honest case, note that the attack in Section 4.1 allows one additional PRF evaluation with probability $1/2^{\ell_1}$. Thus, to prevent this attack, we recommend to raise ℓ_1 to at least κ for the semi-honest case, from the previous works choice $\ell_1 = \lambda + 2 \log n$.

For the malicious case, the adversary can try to overfit OKVS, which finds additional PRF evaluation more efficiently. Here, observe that the possibility of OKVS overfitting depends on the number of H_1 queries, say q , as well as the length ℓ_1 . If we can restrict q less than 2^κ , OKVS overfitting would get harder and then we can utilize more efficient OPRF parameters. This can be simply done by set a sort of timeout for the (corrupt) receiver. To do this, we modify two parts of the protocol, as following.

1. At the beginning of the protocol, the sender samples random salt `salt` and transmit it to receiver. Then, both parties incorporates `salt` into all random oracle inputs. This clearly prevents the malicious receiver prepare H_1 values before the protocol starts.
2. The sender aborts the protocol if the time between sending the salt and receiving the correction matrix back from the receiver exceeds a certain amount of time.

Given that the overfitting attack is more effective than the semi-honest attack, this mitigation technique has an effect of reducing the number of queries for the overfitting attack. We formalize this problem as a new overfitting game as follows.

Definition 6 ($(n, n', q_{\text{on}}, q_{\text{off}})$ -OKVS online overfitting game). *Let (Ecd, Dcd) be an OKVS with parameters chosen to support n items, and let \mathcal{A} be an arbitrary PPT adversary. Run $P \leftarrow \mathcal{A}^{H_1}(1^\kappa, q_{\text{on}})$ where q_{on} is the number of queries to H_1 before outputting P . Define*

$$X' = \{x \mid \mathcal{A} \text{ queried } H_1 \text{ at } x \text{ and } \text{Dcd}(P, x) = H_1(x)\}$$

where \mathcal{A} can query to H_1 less than q_{off} times after P is produced. If $|X'| > n'$, then the adversary wins the $(n, n', q_{\text{on}}, q_{\text{off}})$ -OKVS online overfitting game.

The online query complexity q_{on} should be determined after reviewing a number of factors; such as, computing power of participating parties, timeout time, or network environments. Given that recent Bitcoin's hash rate is about 2^{69} hashes per second,⁴ if those factors are unknown beforehand, $q_{\text{on}} = 2^{96}$ seems a safe choice for not a long timeout time. The offline query complexity q_{off} is the original amount of permitted query, which is usually $O(2^\kappa)$.

Concrete Choice of ℓ_1 . We provide some concrete choices of ℓ_1 that makes OKVS online overfitting game. For that, we first consider n'_1 by the maximum

⁴ This data is retrieved from <https://www.blockchain.com/explorer/charts/hash-rate> in May 2024.

number of allowed evaluations following the PRTY bound given that q_{on} random oracle queries are permitted. Then we let n'_2 be the least integer such that

$$\Pr_{S \leftarrow B(q_{\text{off}}, 2^{-\ell_1})} [S > n'_2] < 2^{-\lambda},$$

which can be rewritten by multiplicative Chernoff bound as follows.

$$\left(\frac{eq_{\text{off}}}{n'_2 2^{\ell_1}} \right)^{n'_2} < 2^{-\lambda}$$

Then, this ℓ_1 is a secure choice of $n' = n'_1 + n'_2$ allowed evaluations. Table 1 summarizes ℓ_1 according to n' and q_{on} .

$\log q_{\text{on}}$ \backslash n'/m	1.5	2	3	4	5
32	111	110	109	108	108
64	134	112	109	108	108
96	226	150	113	111	109
128	322	214	160	141	132

Table 1: The choice of ℓ_1 depending on q_{on} and n' , where m is set to be 1.3×2^{20} , and q_{off} is set to be 2^{128} . For $q_{\text{on}} = 2^{128}$, the offline complexity q_{off} is set to be 0, which is same as the original PRTY bound with $q = 2^{128}$.

In Table 1, a reader might feel that ℓ_1 converges to 108 when n'/m is sufficiently large, whatever q_{on} is. It is because the semi-honest attack allows 2 times more evaluation if ℓ_1 is decreased by a single bit. So, ℓ_1 can be less than 108 if n'/m is large enough; if $n'/m = 1000$, then $\ell_1 = 100$.

On ℓ_1 for the semi-honest model. At the beginning of this section, we recommend to set $\ell_1 = \kappa = 128$ for the semi-honest model. Meanwhile, the choices of ℓ_1 for the malicious model presented in Table 1 for $q_{\text{on}} \leq 2^{96}$ is less than 128. One may think this weird, because the malicious model allows smaller ℓ_1 than the semi-honest model. However, we stress that not only the adversarial model that determines the possible attacks, but the bound n' for the number of PRF evaluations is also an important factor for ℓ_1 : In the semi-honest model, the bound n' is implicitly set by n , whereas the malicious model allows somewhat larger n' such as $c \cdot m$ for $m = (1 + \varepsilon_{\text{okvs}})n$. Finally, it would be possible to extend the definition of semi-honest OPRF to allowing further PRF evaluation n' , which is not explicitly done in our OPRF definition, which enables to set $\ell_1 = 128 - \log n'$ for the semi-honest model.

6.2 Revised Security Proof

We present the revised OKVS-based OPRF construction in Figure 9, and Theorem 2 that specifies the correct conditions of parameters, which reflects the previous works the security flaws.

Correctness. We check that the set $\{H_2(\text{Dcd}(V, x) + w) \mid x \in X\}$ indeed equals to $F(X)$ with the sender's definition

$$F(x) = H_2(x, \text{Dcd}(W', x) + w - \vec{\Delta} \odot C(H_1(x, \text{salt}))).$$

For the readability, we omit salt in H_1 and w for a while, which are common for both parties. Then it holds that from the linearity of Dcd and linear code C

$$\begin{aligned} \text{Dcd}(W', y) - \vec{\Delta} \odot C(H_1(y)) &= \text{Dcd}(V + \vec{\Delta} \odot C(P), y) - \vec{\Delta} \odot C(H_1(y)) \\ &= \text{Dcd}(V, y) - \vec{\Delta} \odot (\text{Dcd}(C(P), y) - C(H_1(y))) \\ &= \text{Dcd}(V, y) - \vec{\Delta} \odot C(\text{Dcd}(P, y)) - C(H_1(y)) \\ &= \text{Dcd}(V, y) - \vec{\Delta} \odot C(\text{Dcd}(P, y) - H_1(y)) \quad (2) \end{aligned}$$

for any $y \in \{0, 1\}^*$. Note that $\text{Dcd}(C(P), y) = C(\text{Dcd}(P, y))$ requires that $\text{row}(y)$ is a vector of \mathbb{B}^{n_c} for every $y \in \{0, 1\}^*$, which is always true when the underlying OKVS is binary. Now, the OKVS correctness property ensures $\text{Dcd}(P, x) = H_1(x)$ for every $x \in X$, then eq. (2) further becomes

$$\begin{aligned} \text{Dcd}(W', x) - \vec{\Delta} \odot C(H_1(x)) &= \text{Dcd}(V, x) - \vec{\Delta} \odot C(\text{Dcd}(P, x) - H_1(x)) \\ &= \text{Dcd}(V, x) - \vec{\Delta} \odot C(H_1(x) - H_1(x)) \\ &= \text{Dcd}(V, x), \end{aligned}$$

which concludes the correctness of our framework.

Security Proof. The underlying proof idea is similar to the previous works [9, 13], whose main idea is to show that $D_y := \text{Dcd}(P, y) - H_1(y)$ has sufficient ($\geq \kappa$) entropy. However, our statement and proof have two distinct points. First, it provides general conditions on the linear code $[n_c, k_c, d]_{\mathbb{B}}$ for arbitrary choice of \mathbb{F} and \mathbb{B} , which correctly applies to both previous constructions. Second, our proof correctly provides the condition on ℓ_1 considering our proposed attack; namely online OKVS overfitting game.

Theorem 2. *Let \mathbb{F} be a field, \mathbb{B} be a subfield of \mathbb{F} , and $\ell_1 > 0$ be an integer that makes $(n, n', q_{\text{on}}, q_{\text{off}})$ -online OKVS overfitting game hard. Let $C = [n_c, k_c, d]_{\mathbb{B}}$ be a \mathbb{B} -linear code where $k_c \cdot \log |\mathbb{B}| \geq \ell_1$ and $d \cdot \log |\mathbb{F}| \geq \kappa$. Then the protocol of Figure 9 securely realizes $\mathcal{F}_{\text{oprf}}$ against $(q_{\text{on}}, q_{\text{off}})$ -malicious adversary in a $\mathcal{F}_{\text{vole}}(\mathbb{F}, \mathbb{B})$ -hybrid model.*

We prove the theorem by the following two lemmas.

Lemma 2. *The the protocol of Figure 9 securely realizes $\mathcal{F}_{\text{oprf}}$ against malicious sender \mathcal{A} who queries to random oracles at most q times.*

Parameters: A finite field \mathbb{F} with a subfield \mathbb{B} , and a $[n_c, k_c, d]_{\mathbb{B}}$ -linear code C . The input set size n_x , and three random oracles $H : \{0, 1\}^* \rightarrow \{0, 1\}^{2\kappa}$, $H_1 : \{0, 1\}^* \rightarrow \{0, 1\}^{\ell_1}$ such that $\ell_1 \leq \log |\mathbb{B}| \cdot k_c$ and $H_2 : \{0, 1\}^* \rightarrow \{0, 1\}^{\ell_2}$. A linear OKVS algorithm pair (Ecd, Dcd) of expansion factor $\varepsilon_{\text{okvs}}$.

Protocol: Upon an input set X from the receiver, the protocol runs as follows:

1. Two parties execute n_c times of (\mathbb{F}, \mathbb{B}) -sVOLE of length $m = (1 + \varepsilon_{\text{okvs}}) \cdot n$. In each i -th execution, the sender obtains $\vec{W}^i \in \mathbb{F}^m$ and $\Delta_i \in \mathbb{F}$ and the receiver obtains $\vec{V}^i \in \mathbb{F}^m$ and $\vec{U}^i \in \mathbb{B}^m$. Two parties let $W, V \in \mathbb{F}^{m \times n_c}$ ($U \in \mathbb{B}^{m \times n_c}$, resp) as matrices of i -th column vector \vec{W}^i, \vec{V}^i (\vec{U}^i , resp).
2. The sender samples $\text{salt} \leftarrow \{0, 1\}^\kappa$, and $w^s \leftarrow \mathbb{F}^{n_c}$, and compute $c^s = H_1(w^s)$. Then it sends salt and c^s to the receiver.
3. The receiver samples $w^r \leftarrow \mathbb{F}^{n_c}$, and computes an OKVS encoding $P \in \{0, 1\}^{m \times \ell_1}$ on the set $\{(x, v_x) : x \in X, v_x = H_1(x, \text{salt})\}$.
4. The receiver applies C on each row vector $\vec{P}_i \in \{0, 1\}^{\ell_1}$ of P by embedding it into \mathbb{B}^{k_c} . Write the resulting matrix by $C(P) \in \mathbb{B}^{m \times n_c}$ whose i -th row is $C(\vec{P}_i) \in \mathbb{B}^{n_c}$.
5. The receiver sends w^r and the correction matrix $U' = C(P) - U \in \mathbb{B}^{m \times n_c}$ to the sender who locally computes $W' = W + \vec{\Delta} \odot U'$.
6. The sender sends w^s to the receiver, who aborts if $c^s \neq H_1(w^s)$. Two parties define $w := w^s + w^r$.
7. The receiver outputs $\{H_2(x, \text{Dcd}(V, x) + w) \mid x \in X\}$ (as $F(X)$).
8. The sender defines the OPRF function $F : \{0, 1\}^* \rightarrow \{0, 1\}^{\ell_2}$ by

$$F(y) := H_2 \left(y, \text{Dcd}(W', y) + w - \vec{\Delta} \odot C(H_1(y, \text{salt})) \right).$$

Fig. 9: Modified OKVS-based OPRF construction.

Proof. The \mathcal{S} interacts with \mathcal{A} as follows:

- \mathcal{S} plays the role of $\mathcal{F}_{\text{vole}}$. \mathcal{S} waits for \mathcal{A} to send $(\vec{W}_i, \Delta_i)_{i \in [n_c]} \in (\mathbb{F}^m \times \mathbb{F})^{n_c}$.
- On behalf of the receiver, \mathcal{S} waits for \mathcal{A} to send salt and c^s . Then, \mathcal{S} sends uniform $w^r \in \mathbb{F}^{n_c}$ and $U' \in \mathbb{B}^{m \times n_c}$. Next, \mathcal{S} waits for \mathcal{A} to send w^s and aborts if $c^s \neq H_1(w^s)$.
- For every queries to H , \mathcal{S} abort if there exists output collision.
- Whenever \mathcal{A} queries $H_2(y, q)$, if $q = \text{Dcd}(W', y) + w - \vec{\Delta} \odot C(H_1(y, \text{salt}))$ and $H_2(y, q)$ has not previously been queried, send y to $\mathcal{F}_{\text{oprf}}$ and programs H_2 to the response. Otherwise, H_2 responses normally.

To prove that this simulation is indistinguishable, consider the following hybrids.

- \mathbf{G}_0 : The same as the real protocol except \mathcal{S} plays the role of $\mathcal{F}_{\text{vole}}$.

- G_1 : \mathcal{S} aborts if there exists output collision in H queries. As H has output length 2κ ,

$$\Pr[G_1 \text{ aborts}] \leq \frac{q^2}{2^{2\kappa}}$$

- G_2 : \mathcal{S} samples U' uniformly instead of computing $U' = C(P) - U$. Since U is chosen uniformly at random, the distribution of \mathcal{A} 's view does not change except when Ecd aborts. Since the probability that Ecd abort is less than $2^{-\lambda}$, the difference from the previous game is negligible:

$$|\Pr[\mathcal{A} \text{ wins } G_1] - \Pr[\mathcal{A} \text{ wins } G_2]| \leq \Pr[\text{Ecd aborts in } G_0] \leq \frac{1}{2^\lambda}$$

- G_3 : When \mathcal{S} samples U' and w^r , abort if there exists (y, σ) such that $H_2(y, \sigma)$ has previously been queried and $\sigma = \text{Dcd}(W', y) + w - \vec{\Delta} \odot C(H_1(y))$. As w^r is chosen uniformly at random, we have

$$\Pr[G_3 \text{ aborts.}] \leq \frac{q}{|\mathbb{F}|^{n_c}} \leq \frac{q}{2^\kappa}$$

Note this hybrid represents why w^r should be sampled by the receiver and sent to the sender.

- G_3 : Whenever \mathcal{A} queries $H_2(y, \sigma)$ after U' is sampled, if $\sigma = \text{Dcd}(W', y) + w - \vec{\Delta} \odot C(H_1(y))$ and $H_2(y, \sigma)$ has not previously been queried, send y to $\mathcal{F}_{\text{oprf}}$ and programs H_2 to the response. Otherwise, H_2 responses normally. As output distribution of $\mathcal{F}_{\text{oprf}}$ is random, the distribution of \mathcal{A} 's view is identical. \square

Lemma 3. *The the protocol of Figure 9 securely realizes $\mathcal{F}_{\text{oprf}}$ against $(q_{\text{on}}, q_{\text{off}})$ -malicious receiver \mathcal{A} .*

Proof. The \mathcal{S} interacts with \mathcal{A} as follows:

- \mathcal{S} plays the role of $\mathcal{F}_{\text{vole}}$. \mathcal{S} waits for \mathcal{A} to send $(\vec{V}_i, \vec{U}_i)_{i \in [n_c]} \in (\mathbb{F}^m \times \mathbb{B}^m)^{n_c}$.
- On behalf of sender, \mathcal{S} sends uniform salt and c^s to \mathcal{A} .
- When \mathcal{A} sends w^r , sample uniform w^s and program $H(w^s) = c^s$.
- When \mathcal{A} sends U' , compute $U' - U = C(P)$. For $H_1(x)$ queries made by \mathcal{A} , \mathcal{S} checks if $\text{Dcd}(C(P), x) = C(H_1(x, \text{salt}))$, \mathcal{S} sends x to $\mathcal{F}_{\text{oprf}}$ and receives it as $F(x)$.
- For each $x \in X$, \mathcal{S} programs $H_2(x, \text{Dcd}(W', x) + w - \vec{\Delta} \odot C(H_1(x)))$ as $F(x)$.

To prove that this simulation is indistinguishable, consider the following hybrids.

- G_0 : The same as the real protocol except \mathcal{S} plays the role of $\mathcal{F}_{\text{vole}}$, so \mathcal{S} waits for \mathcal{A} to send $(\vec{V}_i, \vec{U}_i)_{i \in [n_c]} \in (\mathbb{F}^m \times \mathbb{B}^m)^{n_c}$.
- G_1 : When \mathcal{S} samples uniform salt, \mathcal{S} abort if there exists a prior query from \mathcal{A} to H_1 with form $H_1(\cdot, \text{salt})$. As $|\text{salt}| \geq \kappa$, we have

$$\Pr[G_1 \text{ aborts}] \leq \frac{q_{\text{off}}}{2^\kappa}$$

- G_2 : When \mathcal{A} sends U' and w^r , compute $U' - U = C(P)$. For each of the previous $H_1(x, \text{salt})$ queries made by \mathcal{A} , \mathcal{S} checks if $\text{Dcd}(C(P), x) = C(H_1(x, \text{salt}))$ and if so adds x to X . \mathcal{S} sends X to $\mathcal{F}_{\text{opr}}f$ and receives $\{F(x) \mid x \in X\}$ in response.
- G_3 : \mathcal{S} samples uniform c^s instead of computing H , but programs $H(w^s) = c^s$ after sampling w^s uniformly at random. G_3 aborts if there exists a previous query to H with input collision or there exists a previous query $H_2(x, \sigma)$ such that $\sigma = \text{Dcd}(V, x) + w$. Then, as w^s is chosen uniformly at random from \mathbb{F}^{n_c}

$$\Pr[G_2 \text{ aborts}] \leq \frac{q_{\text{off}}}{\mathbb{F}^{n_c}} \leq \frac{q_{\text{off}}}{2^\kappa}$$

Next, \mathcal{S} programs $H_2(x, \text{Dcd}(V, x) + w) = F(x)$ for $x \in X$ and by the uniformity of $F(x)$, it does not change \mathcal{A} 's view. Note that this hybrid represents why w^s is required.

- G_4 : For each query $H_2(x, \sigma)$, \mathcal{S} add x into X and programs $H_2(x, \sigma) = \mathcal{F}_{\text{opr}}f(x)$ if

$$\begin{cases} \sigma = \text{Dcd}(V, x) + w \\ \text{Dcd}(C(P), x) = C(H_1(x, \text{salt})) \end{cases}$$

then, add x to X and program $H_2(x, \sigma) = \mathcal{F}_{\text{opr}}f(x)$. Similarly, for each query $H_1(x, \text{salt})$, \mathcal{S} add x into X and programs $H_2(x, \text{Dcd}(V, x) + w) = \mathcal{F}_{\text{opr}}f(x)$ if

$$\text{Dcd}(C(P), x) = C(H_1(x, \text{salt})).$$

By the uniformity of $\mathcal{F}_{\text{opr}}f(x)$, this does not change the view of \mathcal{A} . Note that $|X| \leq n'$, by the hardness of $(n, n', q_{\text{on}}, q_{\text{off}})$ -online OKVS overfitting problem.

- G_5 : At the end of protocol, \mathcal{S} samples Δ and aborts if \mathcal{A} ever makes an $H_2(x, \sigma)$ query for $x \notin X$ such that

$$\sigma = \text{Dcd}(W', x) - \vec{\Delta} \odot C(H_1(x, \text{salt})).$$

By (2), for queries $H_2(x, \sigma)$ such that $\text{Dcd}(P, x) \neq H_1(x, \text{salt})$, as there is at least $d \log |\mathbb{F}|$ -bit entropy from $\vec{\Delta}$, we have

$$\Pr[G_5 \text{ abort}] \leq \frac{q}{2^\kappa}.$$

□

6.3 Double Execution of OPRF

This section proposes a novel idea for the extreme case where much tighter $n' \approx m$ is required. Recall that the underlying idea of preventing OKVS overfitting is to limit the number of H_1 queries that the malicious receiver can obtain. The basic idea of what we call double execution of OPRF is to change H_1 by another OPRF, say F , and using the value again to encode OKVS. Precisely, for a given OPRF functionality OPRF_n for n items with variable number of allowed evaluations, the double execution proceeds as follows.

1. The sender and the receiver invoke OPRF_n with $n'' \gg m$ allowed evaluations. The receiver gets at most n'' evaluated items $\{F(x_1), \dots, F(x_{n''})\}$. We will denote the bit-length of H_1 in this phase ℓ_1'' .
2. The sender and the receiver invoke OPRF_n one more time. But the value set V_X at OKVS encoding step at receiver's side becomes $\{F(x) : x \in X\}$ instead of $\{H_1(x) : x \in X\}$. As a result, the receiver gets at most n' evaluated items $\{G(x_{i_1}), \dots, G(x_{i_{n'}})\}$ where $i_k \in [n']$.

The double execution has in fact the same effect of constraining online complexity. In the second phase, since the receiver use $F(x)$ instead of $H_1(x)$, a corrupt receiver can try to overfit an OKVS with only n'' random oracle queries. As ℓ_1 required for tight $n' \approx m$ is so huge in the PRTY bound, the double execution is an economic choice for a tight $n' \approx m$. For example, if a sender and a receiver want to achieve $n' = 1.02m$ in [10] with $q_{\text{on}} = 2^{128}$, $n = 2^{20}$, ℓ_1 should be no less than 5492 bits by PRTY bound. If they use double execution in this case, it is sufficient that $\ell_1'' = 109$ for $n'' = 24m$ and $\ell_1 = 307$ for $n' = 1.02m$. The communication of double execution is $13.2 \approx 5492/(109+307)$ times smaller than the single execution.

7 Better OPRFs from Intermediate Fields

Although the original purpose of Figure 9 was to address both the PRTY construction (with $\mathbb{F} = \text{GF}(2)$) and the RS construction (with $\mathbb{F} = \text{GF}(2^{128})$) simultaneously, it also demonstrates another instantiation with $\mathbb{F} = \text{GF}(2^f)$ for $1 < f < \kappa$, beyond the binary field $\text{GF}(2)$ and the full κ -degree field $\text{GF}(2^\kappa)$. This section discusses these alternative instantiations and reveals more efficient OKVS-based OPRF protocols compared to previous ones.

7.1 Concrete Instantiations

We first consider the instantiation with small-sized \mathbb{F} , for example, $\mathbb{F} = \text{GF}(2^f)$ with $f \leq 10$. In this case, the underlying VOLE can be efficiently realized by the protocol due to [14], referred to as `SoftSpokenVOLE`, which is based solely on Minicrypt assumptions. The computation cost of the VOLE scheme grows exponentially with respect to the parameter f , limiting its practical application when f becomes too large. As a result, the experiments conducted in [14] were restricted to values of f up to 10.

For larger fields \mathbb{F} , the LPN-based VOLE protocol, known as `SilentVOLE` [5], offers a more efficient solution compared to the `SoftSpokenVOLE` protocol. It has a linear computational complexity with respect to the field size, making it suitable for handling large fields without incurring excessive computational costs. However, due to the linear complexity of the VOLE, there is no benefit to using $n_c > 1$. Therefore, we only consider small-sized \mathbb{F} hereafter.

Choice of \mathbb{B} . To complete the instantiation, we need to specify the choice of the subfield degree b that determines $\mathbb{B} = \text{GF}(2^b)$, which can be freely chosen among the divisors of f . Here, we highlight two advantages of taking $b = 1$.

First, the main computation of **SoftSpokenVOLE** consists of \mathbb{B} operations, so taking $b = 1$ makes every computation in the VOLE as bit-operations, resulting in the fastest performance. In fact, the original proposal and its application [2] used $b = 1$ for efficiency. Second, the transmission of the correction matrix $U' \in \mathbb{B}^{m \times n_c}$ (from the receiver to the sender) dominates the communication cost of our OPRF protocol with $m \cdot n_c \cdot \log |\mathbb{B}| = m \cdot n_c \cdot b$ bits. As m is determined by the input set size n and OKVS expansion factor ε_{okvs} , the choice of b only affects $n_c \cdot b$. Considering the general bound $n_c \geq d_c + k_c - 1$ for every linear code and the conditions $k_c \geq \ell_1/b$ and $d_c \geq \kappa/f$ in Theorem 2, we have the inequality $n_c \cdot b \geq \kappa \cdot b/f + \ell_1 - b$. This implies that $b = 1$ allows the minimal communication cost. However, for $b = 1$, the linear code achieving $n_c = d_c + k_c - 1$ (called maximal distance separable code) does not always exist, and that argument cannot assure that $b = 1$ is the best choice. For these reasons, we investigate the known lower bounds of n_c such that the linear code $[n_c, k_c, d_c]_{\mathbb{B}}$ exists in [1] for given k_c and d_c in Table 2, which shows that $b = 1$ provides the minimal $n_c \cdot b$.

$\ell_1 = 109$						$\ell_1 = 150$					
\mathbb{F}	d_c	\mathbb{B}	k_c	n_c	$n_c \cdot b$	\mathbb{F}	d_c	\mathbb{B}	k_c	n_c	$n_c \cdot b$
GF(2 ⁸)	16	GF(2)	109	149	149	GF(2 ⁸)	16	GF(2)	150	192	192
		GF(2 ²)	55	78	156			GF(2 ²)	75	99	198
		GF(2 ⁸)	14	29	232			GF(2 ⁸)	19	34	272
GF(2 ⁶)	22	GF(2)	109	162	162	GF(2 ⁶)	22	GF(2)	150	206	206
		GF(2 ²)	55	86	172			GF(2 ²)	75	107	214
		GF(2 ³)	37	62	186			GF(2 ³)	50	75	225
		GF(2 ⁶)	19	40	240			GF(2 ⁶)	25	46	276
GF(2 ⁴)	32	GF(2)	109	184	184	GF(2 ⁴)	32	GF(2)	150	230	230
		GF(2 ²)	55	99	198			GF(2 ²)	75	120	240
		GF(2 ⁴)	28	59	236			GF(2 ⁴)	38	69	276
GF(2 ²)	64	GF(2)	109	250	250						
		GF(2 ²)	55	145	290						

Table 2: Some minimal possible values of n_c and corresponding $n_c \cdot b$ where $[n_c, k_c, d_c]_{\mathbb{B}}$ can exist. The length $\ell_1 = 109$ (resp., 150) is taken from Table 1 with $q_{\text{on}} = 2^{96}$ with $n' = 5m$ (resp., $2m$).

Meanwhile, the choice of b affects other computational parts. The computation of $F(\cdot)$, particularly the OKVS decoding $\text{Dcd}(W', x)$ or $\text{Dcd}(V, x)$ where $W', V \in \mathbb{F}^{m \times n_c}$, becomes maximal when $b = 1$ since it consists of XOR operations of $(n_c \cdot f)$ -bit strings. Furthermore, two parties need to execute n_c times of (\mathbb{F}, \mathbb{B}) -VOLE, and the number of VOLE instances would be maximal for $b = 1$.

Considering these facts, another choice of b other than 1 could provide a trade-off between computation and communication costs. As verifying the computational benefit of other choices of b requires another extensive experimental

effort, we choose to fix $b = 1$ for its clear communication advantage and leave the investigation of further trade-offs with other choices of $b \neq 1$ for future work.

Communication Cost. This framework consists of two phases of interaction: VOLE, and the transmission of $U' = C(P) - U \in \mathbb{B}^{m \times n_c}$ from the receiver to the sender. Assuming the VOLE communication is negligible (a reasonable assumption due to recent advances in VOLE), the second phase dominates the total communication cost, specifically

$$\text{comm}_{\text{oprf}} = (n_c \log |\mathbb{B}|) \cdot m = (n_c \log |\mathbb{B}|) \cdot (1 + \varepsilon_{\text{okvs}}) \cdot n. \quad (3)$$

Comparison with RS constructions. As the RS construction that combines LPN-based VOLE and OKVS represents the state-of-the-art for OPRF and PSI, we provide some comparisons with it. To recall, the RS construction can be understood as Figure 9 with $\ell_1 = 128$, $\mathbb{F} = \mathbb{B} = \text{GF}(2^{128})$ and the identity linear map $[1, 1, 1]_{\mathbb{B}}$, which requires $128 \cdot m$ bits of communication, as shown in Equation (3). As the value of $n_c \cdot b$ in Table 2 always exceeds 128, the RS construction still incurs a smaller communication cost. However, another important factor is the bound for PRF evaluation n' . The RS construction itself does not have the online hash mitigation, and the PRF bound n' should be calculated with respect to 2^{128} queries, resulting in $n' = 5.6m$ with $m = 1.3n$ (assuming OKVS from [10]). On the other hand, our choice of $\ell_1 = 109$ and 150 comes from $n' = 5m$ and $2m$, respectively, which is smaller than the RS construction's $n' = 5.6m$. In other words, for the original RS construction to achieve $n' = 5m$ or $2m$, their ℓ_1 should be taken larger.

7.2 Performance Evaluation

We present some experimental results to evaluate performance of our newly proposed protocol. Since there is a trade-off between computational and communication complexities, our evaluation was carried out under different network settings with varying the field degree f from 1 to 8 (while fixing the subfield degree by $b = 1$), and the set size $n = 2^{20}$.

For the SoftSpokenVOLE realization, we utilize the implementation of libOTe library [12]. We also choose to employ the OKVS algorithm proposed by [10], whose implementation is publicly available at [16]. Note that this the okvs expansion factor $\varepsilon_{\text{okvs}} \approx 0.3$, so every m below can be regarded as $1.3n$. We remark that RB-OKVS [3] is claimed to have smaller $\varepsilon_{\text{okvs}}$ while having similar performance with [10]. However, we found no public implementation of RB-OKVS, and our internal implementation of RB-OKVS cannot reproduce the numbers in the original paper, so we simply use [10] for experiments. Note that the choice of OKVS might change the absolute numbers in this section, but our main contents are almost independent to the choice of OKVS.

The tests were performed using a machine equipped with 3.50 GHz Intel Xeon E5-1650 v3 (Haswell) CPU and 128 GB RAM, using a single thread. For concrete parameters, we use $\lambda = 40$ bit of statistical security and $\kappa = 128$ bits of

computational security. To simulate various network environments, we employed the `tc` command in conjunction with a local network setup.

ℓ_1	f	n_c	Binary code
109	1	550	RS[50, 19, 32] ₆₄ + [11, 6, 4] ₂
	2	350	RS[50, 19, 32] ₆₄ + [7, 6, 2] ₂
	4	238	RS[34, 19, 16] ₆₄ + [7, 6, 2] ₂
	6	192	RS[32, 22, 11] ₃₂ + [6, 5, 2] ₂
	8	174	RS[29, 22, 8] ₃₂ + [6, 5, 2] ₂
150	1	616	RS[56, 25, 32] ₆₄ + [11, 6, 4] ₂
	2	392	RS[56, 25, 32] ₆₄ + [7, 6, 2] ₂
	4	280	RS[40, 25, 16] ₆₄ + [7, 6, 2] ₂
	6	245	RS[35, 25, 11] ₆₄ + [7, 6, 2] ₂
	8	224	RS[32, 25, 8] ₆₄ + [7, 6, 2] ₂

Table 3: Specification of the binary linear codes for our experiments.

In this evaluation, we focus on the maliciously secure version. We consider two values of ℓ_1 in Table 1 with $q_{\text{on}} = 2^{96}$; $\ell_1 = 109$ for $n' = 5 \cdot m$ for $m \approx 1.3n$, and $\ell_1 = 150$ for $n' = 2 \cdot m$ for $m \approx 1.3n$.

Main Comparison Target. Our main comparison target is the state-of-the-art OPRF protocol [10] in the RS construction (fast version) that using LPN-based VOLE over $\text{GF}(2^{128})$. The performance of those performance strongly depends on the specific choice of the code for LPN. We consider quasi-cyclic LDPC code [5] which we denote below by QC, and a new family of codes named expand-convolute codes proposed by Rindal et al. [11], especially ExConv7x24 (resp. ExConv21x24) implemented in libOTe by EC1 (resp. EC2).

Remark 3. There was another recent proposal of code family called Silver [6] that retains quite aggressive structure to have blazing-fast performance. However, we do not compare with this family, as it turns out to be vulnerable [11].

Concrete Linear Codes. Although Table 2 provides the minimal length of n_c such that $[n_c, \ell_1, \lceil 128/f \rceil]_2$, the concrete construction of such linear code is not known for most cases. Thus, for our experiments, we use the binary linear code constructed by combining a Reed-Solomon (RS) code and another binary code, which method is already used in [9]: To be precise, given ℓ_1 -bit input, we first apply some RS code $\text{RS}[n_{\text{rs}}, k_{\text{rs}}, d_{\text{rs}}]_q$ by embedding ℓ_1 -bit inputs into k_{rs} elements of $\text{GF}(q)$, which outputs n_{rs} elements of $\text{GF}(q)$. After then, we apply $[n_b, \log q, d_b]_2$ code for each $\text{GF}(q)$ element by understanding it as $\log q$ -bits. This implies a binary code $[n_{\text{rs}} \cdot n_b, \ell_1, d_{\text{rs}} \cdot d_b]_2$. Given ℓ_1 and d_c , we exhaustively searched through all possible RS codes and binary linear codes to find the one

$n = 2^{20}, m \approx 1.3n$		Ours		[10]
n'/m		5	2	5.6
ℓ_1		109	150	128
Communication (MB)	$f = 1$	93.56	104.8	
	$f = 2$	59.56	66.71	22.23(QC)
	$f = 4$	40.53	47.68	22.78(EC1)
	$f = 6$	32.71	41.75	22.42(EC2)
	$f = 8$	29.67	38.20	

Table 4: Total communication costs of our OPRF and [10] for $n = 2^{20}$ items. The ‘ n'/m ’ row shows the number of PRF values that malicious receiver can obtain. Our protocols can set $q_{\text{on}} = 2^{96}$ and $q_{\text{off}} = 2^{128}$ thanks to our mitigation, which is not applicable to the previous work.

satisfying $d_{rs} \cdot d_b = d_c$ with the shortest $n_{rs} \cdot n_b$, and the concrete codes are obtained from SageMath. The results are summarized in Table 3.

Remark 4. There would be another construction of linear code $[n_c, \ell_1, d_c]$ shorter than our found RS code & binary code combination; for example, it is known a construction of $[164, 109, 16]_2$ linear code [1] for $\ell_1 = 109$ and $f = 8$ case, which is 10-bit shorter than our construction $[174, 110, 16]_2$. One might try to replace our codes by investigating further shorter linear code, which directly improves the performance without any harm on security.

Communication Costs. Recall that the transmission of the correction matrix $U' \in \mathbb{B}^{m \times n_c}$ dominates the communication cost, precisely $n_c \cdot b \cdot (1 + \epsilon_{\text{okvs}})n_x$ bits. Thus the communication cost would be totally proportional to $n_c \cdot b$, which is n_c in our case, and $b = 128$ in [10]. Table 4 shows the communication cost based on the concrete linear codes found in Table 3, and one can check that indeed the communication cost is exactly proportional to $n_c \cdot b$. We have to say that [10] requires smaller communication than ours, as it has $n_c \cdot b = 128$. However, we would like to remark that the $f = 1$ case is actually corresponds to the original PRTY construction [9] while replacing the subroutines to latest one. In this view, our generalized f choice narrows the gap between [10] protocols and our one: for example $\ell_1 = 109$ (or $n' = 5m$), the communication cost gap between Minicrypt and LPN drops to 4.2x ($f = 1$) to 1.3x ($f = 8$).

Running Time. To check the total running time of OPRF in various network setting, we used 3 settings. The first one is 100Mbps with 100ms rtt (round trip time), and the second one is 1Gbps with 1ms rtt, the last one is 5Gbps with 1ms rtt. These settings are done by using linux `tc` command in local host network. We further note that the public implementation [16] of [10] only supports fixed $\ell_1 = 128$ now, which translates to $n' = 5.6m$. As a fair comparison, we provide the performances for $\ell_1 = 109$ case of $n' = 5m$ in Table 5.

$n = 2^{20}$		100Mbps	1Gbps	5Gbps	Assumption
Ours	$f = 1$	14.2	2.62	2.03	Minicrypt
	$f = 2$	10.0	2.18	1.80	
	$f = 4$	7.787	2.25	1.99	
	$f = 6$	7.783	2.96	2.74	
	$f = 8$	9.37	4.75	4.55	
[10]	QC	5.304	2.32	2.31	dual-LPN
	EC1	4.647	0.990	0.980	
	EC2	5.056	1.629	1.577	

Table 5: Total running time (in second) of our OPRF and [10] on various network settings for $n = 2^{20}$ items. Under the parameters used in this evaluation, our protocol allows at most $n' = 5m$ PRF evaluations, and [10] allows at most $n' = 5.6m$ PRF evaluations.

Table 5 shows OPRF running time with $f = 1, 2, 4, 6, 8$ at our construction and the previous work [10]. The implementation of the previous work is in public [16], so we re-run the code in our evaluation setting for fair comparison. In the fast (5Gbps and 1Gbps) network environments, our $f = 2$ case is even faster than [10] with QC. Since our protocol only requires Minicrypt assumption while QC is not, we might say our protocol is strictly better than it.

In slower network (100Mbps) environment where the communication cost affects a lot, the protocols of [10] are clearly still better. However, we can confirm again the generalized field choice helps to narrow the gap between [10] protocols and our one. Specifically, the $f = 1$ case that corresponds to revised PRTY construction takes 14.2s, which is 2.7 – 3x than [10] protocols, while our $f = 6$ of 7.783s decreases the gap by 1.5 – 1.7x.

Discussion with Breakdown. Table 6 shows the breakdown for of our protocol timing results in Table 5. The ‘Transpose V ’ row is an implementation-specific part, which cannot be seen in Figure 9: The VOLE outputs $V, W \in \mathbb{F}^{m \times n_c}$ are obtained by n_c independent call of VOLE, and they are naturally arranged in a column-wise manner. However, the OKVS decoding understands V (and W) as a length m vector of $n_c \cdot \log |\mathbb{F}|$ -bits, we need to transpose them. As Table 6 shows, such transpose also takes quite a large computational cost.

We finally remark that the first two steps (VOLE and transpose) can be done in offline, before the input set X is determined. In other words, they can be understood as a setup phase before the main OPRF starts. Concretely, for $f = 8$, the first two steps occupy from 77% in the total timings un the 5Gbps network. Thus, we can conclude that our protocol with $f = 8$ would be a nice choice for the situations where some offline computation is allowed, probably even better than [10] protocol with EC-LPN-based VOLE .

f	100Mbps			1Gbps			5Gbps		
	1	4	8	1	4	8	1	4	8
VOLE	0.670	0.822	2.673	0.246	0.430	2.333	0.244	0.414	2.324
Transpose	0.133	0.263	0.664	0.132	0.265	0.675	0.134	0.263	0.667
OKVS Encode	0.356	0.350	0.348	0.356	0.361	0.354	0.352	0.354	0.354
Apply Lin. Code	0.373	0.114	0.100	0.378	0.116	0.100	0.379	0.116	0.101
Send/Recv Corr.	12.14	5.568	4.580	0.995	0.418	0.331	0.404	0.182	0.140
OKVS Decode	0.343	0.491	0.615	0.332	0.485	0.620	0.336	0.485	0.627

Table 6: Breakdown of our protocol timings (in second) in Table 5.

Acknowledgments

The authors would like to thank Peter Rindal for helpful comments and discussions, especially for verifying the early discovery of the flaw and providing clear understanding on the current state of OPRF and PSI protocols.

References

1. Code Tables: Bounds on the parameters of various types of codes (2022), <http://codetables.de/>
2. Baum, C., Braun, L., de Saint Guilhem, C.D., Klooß, M., Orsini, E., Roy, L., Scholl, P.: Publicly Verifiable Zero-Knowledge and Post-Quantum Signatures from VOLE-in-the-Head. In: Handschuh, H., Lysyanskaya, A. (eds.) *Advances in Cryptology – CRYPTO 2023*. pp. 581–615. Springer Nature Switzerland, Cham (2023)
3. Bienstock, A., Patel, S., Seo, J.Y., Yeo, K.: Near-Optimal Oblivious Key-Value Stores for Efficient PSI, PSU and Volume-Hiding Multi-Maps. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 301–318. USENIX Association, Anaheim, CA (Aug 2023), <https://www.usenix.org/conference/usenixsecurity23/presentation/bienstock>
4. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Resch, N., Scholl, P.: Correlated Pseudorandomness from Expand-Accumulate Codes. In: Dodis, Y., Shrimpton, T. (eds.) *Advances in Cryptology – CRYPTO 2022*. Springer Nature Switzerland, Cham (2022)
5. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Rindal, P., Scholl, P.: Efficient Two-Round OT Extension and Silent Non-Interactive Secure Computation. In: *CCS 2019*. pp. 291–308 (2019)
6. Couteau, G., Rindal, P., Raghuraman, S.: Silver: Silent VOLE and Oblivious Transfer from Hardness of Decoding Structured LDPC Codes. In: *CRYPTO 2021*. pp. 502–534. Springer, Cham (2021)
7. Garimella, G., Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: Oblivious key-value stores and amplification for private set intersection. In: *CRYPTO 2021*. pp. 395–425. Springer (2021)
8. Orrù, M., Orsini, E., Scholl, P.: Actively Secure 1-out-of-N OT Extension with Application to Private Set Intersection. In: Handschuh, H. (ed.) *Topics in Cryptology – CT-RSA 2017*. pp. 381–396. Springer International Publishing, Cham (2017)

9. Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: PSI from PaXoS: fast, malicious Private Set Intersection. In: EUROCRYPT 2020. pp. 739–767. Springer (2020)
10. Raghuraman, S., Rindal, P.: Blazing Fast PSI from Improved OKVS and Subfield VOLE. In: CCS 2022. pp. 2505–2517. ACM, New York, NY, USA (2022)
11. Raghuraman, S., Rindal, P., Tanguy, T.: Expand-Convolute Codes for Pseudorandom Correlation Generators from LPN. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023. Springer Nature Switzerland, Cham (2023)
12. Rindal, P.: libOTe: an efficient, portable, and easy to use Oblivious Transfer Library (2022), <https://github.com/osu-crypto/libOTe>
13. Rindal, P., Schoppmann, P.: VOLE-PSI: Fast OPRF and Circuit-PSI from Vector-OLE. In: EUROCRYPT 2021. pp. 901–930. Springer, Cham (2021)
14. Roy, L.: SoftSpokenOT: Quieter OT Extension from Small-Field Silent VOLE in the Minicrypt Model. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13507, pp. 657–687. Springer (2022). https://doi.org/10.1007/978-3-031-15802-5_23, https://doi.org/10.1007/978-3-031-15802-5_23
15. Suzuki, K., Tonien, D., Kurosawa, K., Toyota, K.: Birthday Paradox for Multi-collisions. In: Rhee, M.S., Lee, B. (eds.) Information Security and Cryptology – ICISC 2006. pp. 29–40. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
16. Visa-Research: volepsi: Efficient private set intersection base on vole (2022), <https://github.com/Visa-Research/volepsi>
17. Wagner, D.: A Generalized Birthday Problem. In: Yung, M. (ed.) Advances in Cryptology — CRYPTO 2002. pp. 288–304. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)

A Details for RB-OKVS Overfitting Attack

Case $n' < 2m$. Similarly to PaXoS, we reduce the $(n, \frac{km}{k-1})$ -overfitting game to a k -XOR problem for $k > 2$ as follows. As solving a k -XOR problem of $(d + \ell_1)$ -bit strings costs $O(k2^{\frac{d+\ell_1}{1+\lceil \log k \rceil}})$ time, our attack costs $O(m2^{\frac{d+\ell_1}{1+\lceil \log k \rceil}})$ time including time for querying to the random oracle.

1. Let $Q = \{x_1, \dots, x_q\}$ be an arbitrary subset in $\{0, 1\}^*$. Query all the items in Q to h_0, h_1 , and H_1 . Bucketize Q by h_1 , denoting B_{h_1} the corresponding bucket. Then, there are m buckets, and there are expectedly q/m items per bucket. For simplicity, we assume that $(k-1)|d$ and $(k-1)|m$.
2. For $j \in \{1, k, \dots, m-d-k+2\}$, do the following with initialization $X' \leftarrow \{\}$.
 - (a) Solve a k -XOR problem for $h_0(x)||H_1(x)$ in buckets B_j . If leftmost $k \times (k-1)$ -submatrix has rank strictly less than $k-1$, then repeat this step to find another solution.
 - (b) Gather the proper solutions of the k -XOR problem to X' .
3. For $j \in \{m-d+1, m-d+k, \dots, m-k+2\}$, do the following with initialization $Y \leftarrow \{\}$.
 - (a) Solve a k -XOR problem for $h_0(x)||H_1(x)$ in buckets B_{m-d+1} and let x'_1, \dots, x'_k are solutions.
 - (b) Make a matrix $K \in \mathbb{F}_2^{k \times (m+\ell_1)}$ from $\text{row}(x_i)||H_1(x_i)$.
 - (c) Let K' be a submatrix of K in $\mathbb{F}_2^{k \times (k-1)}$, whose i -th column is equal to the $(j+i-1)$ -th column of K .
 - (d) If $\text{rank}(K') = k-1$, then add x'_1, \dots, x'_k to Y . Otherwise, repeat step 3 for same j .
4. Denote $X' = \{x'_1, \dots, x'_{n'}\}$ where $n' = k \cdot \frac{m-d}{k-1} + k \cdot \frac{d}{k-1} = \frac{km}{k-1}$. Since $\text{rank}(\text{row}(X')) = \text{rank}(\text{row}(X')||H_1(X'))$, there is a solution P' of the linear equation $\text{row}(X') \cdot P' = H_1(X')$.

Case $n' \geq 2m$. Similarly to PaXoS, we reduce the OKVS overfitting problem to a c -multicollision finding problem as follows. As finding c -multicollision of $(d + \ell_1)$ -bit strings costs $O((c!)^{1/c} 2^{\frac{(c-1)(d+\ell_1)}{c}})$ time, our attack costs $O(m2^{\frac{(c-1)(d+\ell_1)}{c}})$ time for small enough c .

1. Let $Q = \{x_1, \dots, x_q\}$ be an arbitrary subset in $\{0, 1\}^*$. Query all the items in Q to h_0, h_1 , and H_1 . Bucketize Q by h_1 , denoting B_{h_1} the corresponding bucket. Then, there are m buckets, and there are expectedly q/m items per bucket.
2. Let $k > 2$. For $j \in \{1, 2, \dots, m-d\}$, do the following with initialization $X' \leftarrow \{\}$.
 - (a) Find a c -multicollision for $h_0(x)||H_1(x)$ from bucket B_j . If the first bit of $h_0(x)$ is 0, find another solution.

- (b) Gather the proper solutions of the problem to X' .
- 3. For $j \in \{m - d + 1, m - d, \dots, m\}$, do the following with initialization $Y \leftarrow \{\}$.
 - (a) Find a c -multicollision for $h_0(x) \| H_1(x)$ in buckets B_{m-d+1} and let x'_1, \dots, x'_c are those solutions.
 - (b) Add proper solutions x'_1, \dots, x'_c to Y .
- 4. If $\text{row}(Y)$ has rank less than d , go back to Step 3. Otherwise, $X' \leftarrow X' \cup Y$.
- 5. Denote $X' = \{x'_1, \dots, x'_{n'}\}$ where $n' = cm$. Since $\text{rank}(\text{row}(\vec{X}')) = \text{rank}(\text{row}(\vec{X}') | H_1(\vec{X}'))$, there is a solution P' of the linear equation $\text{row}(X') \cdot P' = H_1(X')$.