Multi-Client Oblivious RAM with Poly-Logarithmic Communication

Sherman S. M. Chow[1]$^{\star[0000-0001-7306-453X]}$, Katharina Fech[2], Russell W. F. Lai[2], and Giulio Malavolta[3]

[1] The Chinese University of Hong Kong, Hong Kong
[2] Friedrich-Alexander-Uiversität Erlangen-Nürnberg, Germany
[3] UC Berkeley & Carnegie Mellon University, USA
sherman@ie.cuhk.edu.hk, {fech,lai}@cs.fau.de, giulio.malavolta@hotmail.it

Abstract. Oblivious RAM enables oblivious access to memory in the single-client setting, which may not be the best fit in the network setting. Multi-client oblivious RAM (MCORAM) considers a collaborative but untrusted environment, where a database owner selectively grants read access and write access to different entries of a confidential database to multiple clients. Their access pattern must remain oblivious not only to the server but also to fellow clients. This upgrade rules out many techniques for constructing ORAM, forcing us to pursue new techniques. MCORAM not only provides an alternative solution to private anonymous data access (Eurocrypt 2019) but also serves as a promising building block for equipping oblivious file systems with access control and extending other advanced cryptosystems to the multi-client setting.
Despite being a powerful object, the current state-of-the-art is unsatisfactory: The only existing scheme requires $O(\sqrt{n})$ communication and client computation for a database of size n. Whether it is possible to reduce these complexities to $\mathsf{polylog}(n)$, thereby matching the upper bounds for ORAM, is an open problem, *i.e.*, can we enjoy access control and client-obliviousness under the same bounds?
Our first result answers the above question affirmatively by giving a construction from fully homomorphic encryption (FHE). Our main technical innovation is a new technique for cross-key trial evaluation of ciphertexts. We also consider the same question in the setting with N non-colluding servers, out of which at most t of them can be corrupt. We build multi-server MCORAM from distributed point functions (DPF), and propose new constructions of DPF via a virtualization technique with bootstrapping, assuming the existence of homomorphic secret sharing and pseudorandom generators in NC0, which are not known to imply FHE.

Keywords: multi-client oblivious RAM, access control, homomorphic encryption, distributed point function, homomorphic secret sharing

* This work is supported in parts by General Research Funds (CUHK 14209918 and 14210217) and Germany/Hong Kong Joint Research Scheme G-CUHK406/17 of the Research Grants Council, University Grant Committee, Hong Kong, and German Academic Exchange Service under Grant No.: PPP-HK 57391915. The authors would like to thank Brice Minaud and anonymous reviewers for their helpful comments.

1 Introduction

Oblivious RAM (ORAM) [22] allows random accesses to physical memory locations without revealing the logical read/write access patterns. The original motivation considers a software accessing the local memory, where the latter is modeled as a machine that can only perform read and write operations but no computation (known as the "balls and bins" model). Later, ORAM was also considered in a network setting, where a client wishes to obliviously access its data outsourced to a remote server, where computation might be allowed. Besides direct applications in local and remote storage, ORAM techniques have been shown useful for many other cryptographic goals.

In a realistic setting, a database can be accessed by hundreds of mutually untrusted clients. The security of ORAM or even its parallel variant (OPRAM) [3,13] becomes insufficient as all clients (processors in the same machine in OPRAM) share the same secret key. To remedy this, Maffei *et al.* [29] considered multi-client ORAM (MCORAM), which aims to capture the following natural scenario: A database owner encodes an array of data M and outsources the encoded database to a server. Clients can dynamically join the system and request access rights to individual entries of M. After the permission is granted, a client can obliviously perform random access to the permitted entries of M, without communicating with the database owner or other clients.

For privacy, MCORAM expects two strengthened requirements against an adversary who can *corrupt an arbitrary subset of clients* and the server:
- Read and write accesses are anonymous.
- Read and write accesses are indistinguishable, except when the adversary has read access to the address being written.

Integrity is another interesting security feature needed in a multi-client scenario – legitimately written entries should be retrievable by any permitted clients and cannot be overwritten by malicious clients, assuming an honest server.

After three decades of development, the complexities of ORAM schemes are well-understood. Unfortunately, many techniques for constructing ORAM break down completely when the client can be corrupt. This forces us to pursue new techniques in building MCORAM, regardless of the many ORAM constructions.

The only (fully-oblivious) MCORAM by Maffei *et al.* [30] requires $O(n)$ server computation and $O(\sqrt{n})$ communication and client computation. They also show $\Omega(n)$ server computation is *necessary* (in the balls-and-bins model), in contrast to $\mathsf{polylog}(n)$ computation of ORAM. For communication, no non-trivial lower bound for MCORAM is known, while the upper bounds for ORAM and MCORAM are $\mathsf{polylog}(n)$ and $O(\sqrt{n})$, respectively. The inherent complexities of MCORAM are still poorly understood. We are thus motivated to ask:

Is multi-client ORAM with $\mathsf{polylog}(n)$ ***communication possible?***

Our Results. We answer the above question affirmatively. Our main contribution is a single-server MCORAM construction with $O(\log n)$ communication and client computation, and $O(n)$ server computation (omitting factors of $\mathsf{poly}(\lambda)$).

Table 1. Comparison of MCORAM schemes for storing n messages in the following criteria: security against malicious clients (MC), support of multiple data owners (MD), security against t out of N corrupt servers, server computation, client computation, and communication of the access protocol (factors of $\mathsf{poly}(\lambda)$ are omitted.)

Scheme	MC	MD	(t, N)	S Comp	C Comp	Comm
[29]	✘	✘	$(1,1)$	$O(\log n)$	$O(\log n)$	$O(\log n)$
[30]	✓	✓	$(1,1)$	$O(n)$	$O(\sqrt{n})$	$O(\sqrt{n})$
This work	✓	✓	$(1,1)$	$O(n)$	$O(\log n)$	$O(\log n)$
This work	✓	✓	$(N'-1, {N'}^2)$, $N' \in \{2,3,4\}$	$O(n)$	$\mathsf{polylog}(n)$	$\mathsf{polylog}(n)$

This scheme relies on a new usage of key-indistinguishable FHE, in which ciphertexts encrypted under an unknown key are evaluated under non-matching keys. When instantiated with a rate-1 FHE [8,19], the communication complexity is optimal ($\log n$) up to an additive fixed polynomial factor. In other words:

Theorem 1 (Informal). *Assuming FHE, there exists a multi-client ORAM scheme with poly-logarithmic communication complexity.*

We also consider the setting with multiple (non-colluding) servers, in which we propose an N^2-server MCORAM scheme resilient against the corruption of t servers, with $\mathsf{polylog}(n)$ communication and client computation. The scheme assumes the existence of a (t, N) distributed point function (DPF) [21], where N is the number of parties and t is the corruption threshold. We then show new constructions of DPFs for parameters $(t, N) \in \{(2,3),(3,4)\}$, respectively, assuming homomorphic secret sharing (HSS) and constant-depth pseudorandom generators (PRGs), which are not known to imply FHE. Together with the existing $(1,2)$-DPF [21], we show the following theorem.

Theorem 2 (Informal). *Assuming HSS and PRG in NC0, there exist multi-client* $\{4, 9, 16\}$*-servers ORAM schemes with poly-logarithmic communication complexity, resilient against the corruption of* $\{1, 2, 3\}$ *servers respectively.*

As summarized in Table 1, we made clear contributions in communication and client computation complexities. One may further ask for an even better construction as (i) the computation of the servers is linear in the database size and (ii) the client storage is proportional to the number of entries with access granted. While the former is inherent to some extent (as shown in [30]) and the latter appears to be natural for fine-grained access control allowing $O(2^n)$ possible policies for each user, we show how to reduce the client storage by constrained PRFs [2]. For simple access structures (such as prefix predicates), known constrained PRFs (*e.g.*, [24]) do not add any extra assumption.

2 Technical Overview

2.1 MCORAM with Poly-Log Communication: Initial Attempts

A first attempt to construct MCORAM with poly-logarithmic communication is to extend an ORAM with the same complexity. Simply sharing the same ORAM secret key among all clients (*e.g.*, [25]) fails. The *secret state* kept by each client is the root issue. For obliviousness against the server and fellow clients, they must be kept confidential from others. To ensure consistency of the operations across all clients, they must be correlated. These *contradicting* requirements seem to forbid the adoption of many ORAM techniques. Another idea is to secret-share the ORAM secret key to all clients, and emulate the ORAM accesses using secure multi-party computation. This requires interactions between many clients for each access and is clearly undesirable when the number of clients is large.

We note that a database can be privately accessed without a persistent secret client state in (single-server) private-information retrieval (PIR) [14], in which a *stateless* client can *read* an entry while hiding its address. For the discussion below, it is useful to recall the standard FHE-based PIR scheme, which achieves poly-logarithmic communication. Recall that FHE allows homomorphic evaluations of any circuits over ciphertexts. To read the entry $M[a]$ of a database M at address a, the client samples a fresh FHE key pair $(\mathsf{pk}, \mathsf{sk})$ and sends $(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, a))$ to the server. The server homomorphically evaluates the following circuit Read_M parameterized by M over $\mathsf{Enc}(\mathsf{pk}, a)$:

$\mathsf{Read}_M(\mathsf{addr})$: Return $M[\mathsf{addr}]$.

This results in a ciphertext encrypting $M[a]$ to be sent to the client.

We can extend a PIR scheme to the multi-client setting and yield a read-only MCORAM. More concretely, the data owner encrypts each database entry with a different key. Granting read access means delegating the decryption key of the corresponding address. To read, recall that PIR clients are stateless, the client first performs PIR, and then decrypts the retrieved encrypted entry locally.

Challenge: Write Access. Towards supporting write access, a rough idea is as follows. First, each database entry $M[a]$ is encrypted under FHE, so the server cannot just see which entries have changed after a write access. Next, when writing data m^* to address a^*, the client encrypts its update instruction (a^*, m^*) using FHE, so that the server could "somehow" update the database entries homomorphically by evaluating a Write function over the ciphertexts of (a^*, m^*) and (each entry of) M. This raises the question of – Under which key should (1) each entry $M[a]$, and (2) the update instruction (a^*, m^*) be encrypted?

Using the same key across all $M[a]$ fails as we discussed – all clients need to hold the same decryption key to access their data. Now we need to encrypt each $M[a]$ under a key pk_a independently generated for each a. With $O(n)$ communication and client computation, the client can just create n FHE-ciphertexts, each using a different key, and sends them to the server. With the poly-logarithmic constraint, we face a *dilemma*: Either the client informs the server about (a^*, pk_{a^*}) so that the latter knows which ciphertext it should update, which violates obliviousness; or the server would need to somehow evaluate Write over a^*, m^*, and $M[a]$, where the first two are encrypted under pk_{a^*}, and the last is under pk_a, for $a \in [n]$, and then it is unclear if correctness would hold. Multi-key FHE does not seem to be useful in this context because its homomorphic evaluation results in a ciphertext under a new combined key, which creates a complicated key-management problem and suffers from the problem of high interaction similar to the secure multi-party computation solution.

2.2 FHE-based Construction

Our insight into resolving the dilemma is that some meaningful operations can actually be done over FHE ciphertexts encrypted under *different* keys. Specifically, we introduce a *cross-key trial evaluation* technique that interprets a ciphertext as one encrypted under a possibly mismatching key.[4] Below, we illustrate our technique with a simplified setting that is sufficient to capture the essence.

Cross-Key Trial Evaluation. Recall that the server database stores $\big(a, \mathsf{Enc}(\mathsf{pk}_a, M[a])\big)$ for $a \in [n]$. To write, a client sends the encrypted instruction $\big(\mathsf{Enc}(\mathsf{pk}_{a^*}, a^*), \mathsf{Enc}(\mathsf{pk}_{a^*}, m^*)\big)$ to the server, which evaluates for each $a \in [n]$ the following simplified writing circuit, parameterized by a, over $\mathsf{Enc}(\mathsf{pk}_{a^*}, a^*)$ and $\mathsf{Enc}(\mathsf{pk}_{a^*}, m^*)$ from the client, and $\mathsf{Enc}(\mathsf{pk}_a, M[a])$ from the server storage, by treating as if all of them were created under pk_a:

$\mathsf{SimpleWrite}_a(\mathsf{addr}, \mathsf{data}', \mathsf{data})$: If $\mathsf{addr} = a$, return data'; else return data.

For each $a \in [n]$, the server overwrites the a-th ciphertext it stored with the ciphertext output by evaluating $\mathsf{SimpleWrite}_a$. Let us examine what happens depending on whether a matches a^* from the update instruction. If $a = a^*$, all three ciphertexts are encrypted under the same key; the server would get a ciphertext of m^* under pk_{a^*}, *i.e.*, $M[a^*]$ is correctly overwritten with m^*.

If $a \neq a^*$, it seems paradoxical that this evaluation gives us anything meaningful since there is no correctness guarantee when homomorphic evaluations are performed under a *wrong public key* $\mathsf{pk}_{a^*} \neq \mathsf{pk}_a$. However, as a matter of fact, the homomorphic evaluation still proceeds as if everything is encrypted under pk_a. Namely, it interprets its input, particularly the first ciphertext $\mathsf{Enc}(\mathsf{pk}_{a^*}, a^*)$, as if it is encrypted under pk_a. With this treatment, it is very unlikely that $\mathsf{Enc}(\mathsf{pk}_{a^*}, a^*)$ is also a ciphertext of a under pk_a. More precisely, $\mathsf{Dec}(\mathsf{sk}_a, \mathsf{Enc}(\mathsf{pk}_{a^*}, a^*))$ should be "independent" of a (we will revisit this shortly), and therefore the check $\mathsf{addr} = a$ would most likely fail. Then, by the correctness of FHE, the evaluation would result in a new ciphertext encrypting $\mathsf{data} = M[a]$ under pk_a, *i.e.*, entries $M[a]$ with $a \neq a^*$ remain unchanged.

The critical insight here is that the random outcomes of operating on a "wrong" ciphertext, with overwhelming probability, "match" with the desired behavior we expect as if *cross-key evaluation* is possible. Note that after each write operation, the entries are left in a consistent state, *i.e.*, each entry $M[a]$ is still encrypted under pk_a, and the database size stays the same. For this to be true, our FHE scheme must satisfy a strong variant of correctness, where the

[4] This technique is reminiscent of *decrypting* a *random* string interpreted as an FHE ciphertext in the surprising result of Canetti, Lombardi, and Wichs [12], which constructs non-interactive zero-knowledge from any circular-secure FHE.

evaluation algorithm must be well-defined and correct over the entire ciphertext space (and not necessarily in the support of a particular public key). In Section 4.2, we show how to generically transform any FHE scheme to satisfy this notion, provided that it meets some weak structural requirements.

Finally, the FHE scheme here needs to be *key-private*, *i.e.*, ciphertexts under different keys are indistinguishable. Fortunately, essentially all known FHE schemes are key-private, as their ciphertexts are typically indistinguishable from uniformly sampled elements from the ciphertext space.

Achieving Integrity and a Formal Reduction. The above approach can provide writing functionality, *but not integrity* as everyone can encrypt using the keys pk_a. Furthermore, we relied on the heuristic that $\mathsf{Dec}(\mathsf{sk}_a, \mathsf{Enc}(\mathsf{pk}_{a^*}, a^*)) \neq a$ with high probability, which is difficult to guarantee formally.

We propose a technique that resolves both issues simultaneously using any signature scheme Σ. Clients with writing rights to a are granted an address-dependent signing key sk_a^Σ. Instead of encrypting (a^*, m^*), the client computes $\mathsf{Enc}(\mathsf{pk}_{a^*}, \sigma^*)$ and $\mathsf{Enc}(\mathsf{pk}_{a^*}, m^*)$, where σ^* is a signature of (r, m^*) under $\mathsf{pk}_{a^*}^\Sigma$, and r is a random nonce chosen by the server for each access. Correspondingly, the server homomorphically evaluates for each a the circuit $\mathsf{Write}_{\mathsf{pk}_a^\Sigma, r}$, parameterized by $(\mathsf{pk}_a^\Sigma, r)$, over the ciphertexts of $\mathsf{Enc}(\mathsf{pk}_{a^*}, \sigma^*)$, $\mathsf{Enc}(\mathsf{pk}_{a^*}, m^*)$, and $\mathsf{Enc}(\mathsf{pk}_a, M[a])$, again as if they are all ciphertexts under pk_a:

$\mathsf{Write}_{\mathsf{pk}_a^\Sigma, r}(\mathsf{sig}, \mathsf{data}', \mathsf{data})$: If sig is a valid signature of (r, data') under pk_a^Σ, return data'; else return data.

With a similar argument as above, $M[a]$ would be overwritten by m^* if $a^* = a$ and σ^* is a valid signature, which can only be generated by clients having writing rights to a^*. Unlike using $\mathsf{SimpleWrite}_a$, we can further argue about the converse without relying on heuristics. Concretely, if $a \neq a^*$ but $\mathsf{Dec}(\mathsf{sk}_a, \mathsf{Enc}(\mathsf{pk}_{a^*}, \sigma^*))$ is a valid signature of $\big(r, \mathsf{Dec}(\mathsf{sk}_a, \mathsf{Enc}(\mathsf{pk}_{a^*}, m^*))\big)$ under pk_a^Σ, we can extract a signature forgery with respect to the verification key pk_a^Σ, violating the unforgeability of the signature scheme. Consequently, it holds that when $a \neq a^*$, $M[a]$ would not be overwritten except with negligible probability.

Applications. The above technique can be generalized to enable *(key-dependent) conditional evaluations* of FHE ciphertexts, with the condition depends on not only the messages encrypted within but also the keys used to generate the ciphertexts. This feature is useful in (outsourced) access-control applications such as an *"oblivious whitelisting firewall"* that only allows incoming ciphertexts encrypted under one of the whitelisted keys to pass through without the firewall keeping any secret key.

Reducing Secret Key Size. So far, we have assumed that the data owner generates address-dependent secret keys, and grants clients reading and writing rights by delegating the keys for the corresponding addresses. In the worst case, data owner and client keys are of size linear in the size of the database.

A common technique to reduce the data-owner key size is to generate those address-dependent secret keys by a pseudorandom function (PRF). Towards reducing the client key size, a constrained PRF (cPRF) can be used. Recall that cPRF can create a constrained key K_X that constrains the PRF key K within some subset X of the domain. Given K_X, one can evaluate the PRF on all inputs $x \in X$, while the PRF values of all $x \notin X$ remain pseudorandom. That means the data owner can delegate to the clients cPRF keys that allow derivation of the address-dependent secret keys. If the cPRF keys are succinct, *e.g.*, of size sublinear in the size of X, the client key size is also short. For example, the well-known PRF construction by Goldreich, Goldwasser, and Micali [24] is a cPRF for prefix constraints with logarithmic-size keys.

On Sublinear Server Computation. The focus of our work is to minimize the communication complexity of the protocol. We note that recent works [11,7] have investigated the possibility of sublinear server computation (with preprocessing) in PIR (essentially a read-only MCORAM with no access control) in the *single-client* setting, and have proposed a solution based on new hardness assumptions on permuted Reed-Solomon codes. They also consider the *public-key* setting, which does not require any secret state to read the database, *i.e.*, multiple clients are allowed to query the database obliviously. Unfortunately, the only proposed solutions build on a strong notion of virtual black-box obfuscation. We consider constructing an MCORAM with sublinear server computation (from standard assumption) as a fascinating open problem.

2.3 DPF-based Multi-Server Construction

The scheme described above resolves the open question of communication efficiency for MCORAM using FHE schemes, which are yet to become efficient in a practical sense, and are only known to be realizable from lattices. Towards finding more practical solutions, to broaden the spectrum of assumptions, and to get a larger variety of MCORAM schemes, we turn our attention to the multi-server setting, in which we leverage the non-collusion between different servers. We restrict to the three-message setting where *the servers do not talk to each other*. This motivates the non-collusion assumption and rules out trivial constructions.[5]

In this direction, we rely on another tool known as distributed point functions (DPF), which were shown to be useful in constructing PIR and in complexity theory [21], as well as private queries on public data [32]. A DPF allows a client

[5] In the three-message setting, the accessing client sends one message to each of the servers, each server sends one message back to the client, and the client sends one final message back to each server. Thus, the servers cannot communicate with each other (even through the client) in coming up with the responses to the client. Not letting the servers communicate also ruled out any straightforward adaption that evaluates an ORAM under the hood of secure two/multi-party computation.

to split a given point function into keys $(k_1, \ldots, k_N)$. Given k_i, one can locally evaluate the shared function at some input point to obtain a value z_i. Computing $z_1 + \cdots + z_N$ reconstructs the function output at the evaluated point. If the point function is *hidden* even if t-out-of-N shares are leaked, we call it a (t, N)-DPF. The main efficiency measure for a DPF is the size of the shares, which can be as small as $\log n$, where n is the size of the truth table of the point function.

We are going to build DPFs for new values of (t, N) not achieved before. In particular, the existing query system [32] was only instantiated by $(1, 2)$-DPF.

From DPF to Multi-Server MCORAM. There is a folklore N^2-server ORAM construction (a.k.a. distributed ORAM [9]) assuming only a (t, N)-DPF. Using a DPF with $\mathsf{polylog}(n)$ communication, the construction achieves $\mathsf{polylog}(n)$ communication. While its server computation complexity is $O(n)$, it has been shown to outperform other optimized competitors in practice [16]. More importantly, we observe that we can adopt this DPF-based scheme to the multi-client setting in a relatively simple manner.

On a very high level, the construction arranges a set of N^2 servers in a square matrix according to some (*e.g.*, lexicographical) ordering. The database M is split into N shares such that $\bar{M}_1 + \cdots + \bar{M}_N = M$, and all servers belonging to the i-th row are given the i-th share $\bar{M}_i$. Clients can read the a-th location $M[a]$ by sharing a point function (which evaluates to a bit-string with the a-th bit being 1) to each server in some i-th row. The responses from a server allow the client to decode the i-th share of $M[a]$. Repeating this for all N rows, the client could recover all shares and hence $M[a]$. Writing can be done similarly, except that shares of the DPF are distributed row-wise to keep the share of the databases consistent (see Section 7 for more details).

New DPF Constructions. With the generic transformation, we can focus on constructing DPFs. The *only known* DPF with (poly)logarithmic-size shares from non-lattice assumptions is due to Boyle *et al.* [4]. They show how to construct a $(1, 2)$-DPF with logarithmic-size shares, assuming only the existence of PRGs, which is equivalent to the existence of one-way functions. This yields a $(1, 4)$-MCORAM resilient against a single server.

For improving resilience against a higher number of faulty servers, we investigate new constructions of DPFs with different parameters. In this work, we build a $(2, 3)$-DPF and a $(3, 4)$-DPF with poly-logarithmic communication. These new constructions give us a $(2, 9)$-MCORAM and a $(3, 16)$-MCORAM, respectively.

The design blueprint is as follows. We start with a crucial observation that the evaluation algorithm of the existing $(1, 2)$-DPF [4] can be run in an NC1 circuit by instantiating the underlying PRG appropriately. Our key insight into increasing the number of parties is *a virtualization technique* for emulating the execution of the DPF evaluation algorithm of each party by 2 servers. To realize such *bootstrapping*, we leverage another tool called homomorphic-secret sharing [6]. By applying our techniques to one or two parties, we obtain a $(2, 3)$-DPF and a $(3, 4)$-DPF, respectively. Both schemes rely on a PRG that can be computed in

NC0 (*e.g.*, Goldreich PRG [23]) and either the decisional Diffie-Hellman (DDH) or the decisional composite residuosity (DCR) assumption.

3 Related Work

ORAM has been extensively studied for more than three decades, but mostly in the single client setting, with drastically different research challenges compared to the multi-client setting. For example, S^3ORAM [27] is a *single-client* ORAM that splits the server-side computation across 3 servers via secure multiparty computation, which we aim to avoid. Recent works [3,13] considered how to preserve obliviousness when a large number of clients access the database in parallel, without considering security against *malicious* clients or *access control.* These works require the clients to *synchronize* with each other and periodically *interact* with the data owner, which is not needed by our MCORAM constructions.

ORAM and similar cryptographic techniques such as private information retrieval (PIR) [14,28] have been utilized in building oblivious file systems (*e.g.*, TaoStore [31] and prior works cited by [30,31]). These systems do not support access control, and their obliviousness does not hold against malicious clients. (Also see [30, Table 1].) Oblivious transfer (OT) can be considered as an ORAM without writing. Camenisch *et al.* [10] proposed OT with access control. Seeing a valid zero-knowledge proof of the client credential, the "server" helps the client *decrypt* one (randomized) entry of the encrypted database previously sent to the client. Since the decryption key is needed, the *data owner should remain online.*

Also relying on zero-knowledge proofs, group ORAM [29] allows the client to access the database according to a predefined policy without any interaction with the data owner. Yet, the obliviousness does not hold against *malicious* clients.

Blass, Mayberry, and Noubir [1] proposed "multi-client ORAM" in a model different from ours, in which all the clients trust each other. Their focus is security against a server that is actively malicious and may rewind the state information shared by multiple clients (and stored by the server).

A recent work of Hamlin *et al.* [26] considered a closely related problem called private anonymous data access (PANDA), yet with some crucial differences. PANDA can be considered as combining the best of PIR and ORAM. It focuses on achieving sublinear server computation, leveraging assumptions such as only t out of the N clients can be corrupt for some *predefined threshold* t, and the set of clients are fixed at setup. In contrast, MCORAM allows any subset of the clients to be corrupt, and clients can dynamically join the system. All PANDA schemes have both communication and computation complexities *scale multiplicatively* in t. One of their schemes (Secret-Writes PANDA) achieves the closest functionality aimed by MCORAM. However, writing is append-only, meaning that their server storage grows linearly in the total number of writes performed by all clients. Reads and writes are also distinguishable. While one could hide the access type by performing dummy reads and writes, the append-only nature makes *the server storage grows linearly in the number of reads and writes.* In short, MCORAM with $\mathsf{polylog}(n)$ communication provides a better alternative with no reliance on the client corruption threshold for security or communication efficiency.

4 Preliminaries

Let PPT denote probabilistic polynomial time. The security parameter is denoted by $\lambda \in \mathbb{N}$. We say that a function $\mathsf{negl}(\cdot)$ is negligible if it vanishes faster than any inverse polynomial. We write the set $\{1, \ldots, N\}$ as $[N]$.

4.1 Constrained Pseudorandom Functions

A constrained PRF (cPRF) [2] is a PRF equipped with the additional algorithms $\mathsf{Constrain}$ and cEval. Let $\mathcal{X}$ be the domain of the PRF. For any subset $X \subseteq \mathcal{X}$, $\mathsf{Constrain}$ produces a constrained key K_X from the secret key K. Given K_X, cEval can evaluate the PRF over any input $x \in X$, yet the PRF values for $x' \notin X$ remain pseudorandom. We focus on polynomial-size domains, so the membership $x \in X$ for any $X \subseteq \mathcal{X}$ can be checked in polynomial time.

Definition 1 (Constrained Pseudorandom Functions). *A constrained pseudorandom function family with domain $\mathcal{X}$ and range $\mathcal{Y}$ is defined as a tuple of* PPT *algorithms* $(\mathsf{KGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{cEval})$ *such that:*

$\underline{\mathsf{KGen}(1^\lambda)}$: *On input the security parameter 1^λ, the key generation algorithm returns a secret key K.*

$\underline{\mathsf{Eval}(K, x)}$: *On input the secret key K and a value $x \in \mathcal{X}$, the* deterministic *evaluation algorithm returns a (pseudorandom) value $y \in \mathcal{Y}$.*

$\underline{\mathsf{Constrain}(\mathsf{sk}, X)}$: *On input the secret key and a set $X \subseteq \mathcal{X}$, the constrain algorithm returns a constrained secret key K_X.*

$\underline{\mathsf{cEval}(K_X, x)}$: *On input a constrained key K_X and a value $x \in \mathcal{X}$, the* deterministic *constrained evaluation algorithm returns a value $y \in \mathcal{Y}$ or $\perp$.*

We only require a cPRF to satisfy weak selective-input variants of correctness and pseudorandomness, where the adversary first commits to a set ChSet before given access to the evaluation and constrain oracles. The adversary promises not to query the oracles over inputs which has any intersection with ChSet.

Definition 2. *A constrained PRF* cPRF *with domain $\mathcal{X}$ and range $\mathcal{Y}$ is said to be selective-input correct if, for all* PPT *algorithms $\mathcal{A}$, it holds that*

$$\Pr\left[\mathsf{Correctness}_{\mathcal{A},\mathsf{cPRF}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Correctness}_{\mathcal{A},\mathsf{cPRF}}$ *is defined in Fig. 1.*

Definition 3. *A constrained PRF* cPRF *with domain $\mathcal{X}$ and range $\mathcal{Y}$ is said to be selective-input pseudorandom if, for all* PPT *algorithms $\mathcal{A}$, it holds that*

$$\left|\Pr\left[\mathsf{Pseudorandom}^0_{\mathcal{A},\mathsf{cPRF}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Pseudorandom}^1_{\mathcal{A},\mathsf{cPRF}}(1^\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Pseudorandom}^b_{\mathcal{A},\mathsf{cPRF}}$ *is defined in Fig. 1.*

```
$\mathsf{Correctness}_{\mathcal{A},\mathsf{cPRF}}(1^\lambda)$
$\mathsf{ChSet} \leftarrow \mathcal{A}(1^\lambda)$
$K \leftarrow \mathsf{KGen}(1^\lambda)$
$(x, X) \leftarrow \mathcal{A}^{\mathsf{Eval}\mathcal{O},\mathsf{Constrain}\mathcal{O}}(1^\lambda)$
$K_X \leftarrow \mathsf{Constrain}(K, X)$
$b_0 := (x \in X)$
$b_1 := (\mathsf{Eval}(K, x) \neq \mathsf{cEval}(K_X, x))$
**return** $b_0 \wedge b_1$

$\mathsf{Eval}\mathcal{O}(x)$
**ensure** $x \notin \mathsf{ChSet}$
$y := \mathsf{cPRF.Eval}(K, x)$
**return** $y$
```

```
$\mathsf{Pseudorandom}^b_{\mathcal{A},\mathsf{cPRF}}(1^\lambda)$
$\mathsf{ChSet} \leftarrow \mathcal{A}(1^\lambda)$
$K \leftarrow \mathsf{KGen}(1^\lambda)$
$f \leftarrow \mathcal{Y}^{\mathcal{X}}$ // the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$
$Y_0 := \{f(x) : x \in \mathsf{ChSet}\}$
$Y_1 := \{\mathsf{cPRF.Eval}(K, x) : x \in \mathsf{ChSet}\}$
$b' \leftarrow \mathcal{A}^{\mathsf{Eval}\mathcal{O},\mathsf{Constrain}\mathcal{O}}(Y_b)$
**return** $b$

$\mathsf{Constrain}\mathcal{O}(X)$
**ensure** $X \cap \mathsf{ChSet} = \emptyset$
$K_X := \mathsf{cPRF.Constrain}(K, X)$
**return** $K_X$
```

Fig. 1. Correctness and Pseudorandomness Experiments for Constrained PRFs

4.2 Fully Homomorphic Encryption

Definition 4 (Fully Homomorphic Encryption). *Let $\mathcal{K} = \mathcal{K}_\lambda$ be a secret key space, $\mathcal{M} = \mathcal{M}_\lambda$ be a plaintext space, and $\mathcal{C} = \mathcal{C}_\lambda$ be a ciphertext space. For each $n \in \mathbb{N}$, let $\mathbb{C}_n$ be the set of all polynomial-size circuits from $\mathcal{M}^n \to \mathcal{M}$. A homomorphic encryption scheme is defined as a tuple of* PPT *algorithms below.*

$\underline{\mathsf{KGen}(1^\lambda)}$: *On input the security parameter $\lambda \in \mathbb{N}$, this key generation algorithm returns a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$ where $\mathsf{sk} \in \mathcal{K}$.*

$\underline{\mathsf{Enc}(\mathsf{pk}, m)}$: *On input pk and a message $m \in \mathcal{M}$, this encryption algorithm returns a ciphertext $c \in \mathcal{C}$.*

$\underline{\mathsf{Dec}(\mathsf{sk}, c)}$: *On input $\mathsf{sk} \in \mathcal{K}$ and a ciphertext $c \in \mathcal{C}$, this decryption algorithm returns the plaintext $m \in \mathcal{M}$.*

$\underline{\mathsf{Eval}(\mathsf{pk}, \mathsf{C}, (c_1, \ldots, c_n))}$: *On input a public key pk, a polynomial-size circuit $\mathsf{C} \in \mathbb{C}_n$, and a set of ciphertexts $(c_1, \ldots, c_n) \in \mathcal{C}^n$ for some $n \in \mathbb{N}$, this evaluation algorithm returns an evaluation output $c' \in \mathcal{C}$.*

Fix $\lambda \in \mathbb{N}$. For each $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}(1^\lambda)$, we recursively define $\mathcal{C}_{\mathsf{pk}} :=$

$$\left\{ c : \begin{array}{c} \big(\exists m \in \mathcal{M} \text{ s.t. } c \in \mathsf{Enc}(\mathsf{pk}, m)\big) \\ \vee \Big(\exists n \in \mathbb{N}, \mathsf{C} \in \mathbb{C}_n, (c_1, \ldots, c_n) \in \mathcal{C}^n_{\mathsf{pk}} \text{ s.t. } c \in \mathsf{Eval}(\mathsf{pk}, \mathsf{C}, (c_1, \ldots, c_n))\Big) \end{array} \right\}$$

to be the space of "well-formed" ciphertexts under the key pk, *i.e.*, all ciphertexts produced by $\mathsf{Enc}(\mathsf{pk}, \cdot)$ and $\mathsf{Eval}(\mathsf{pk}, \cdot, \cdot)$. Apparently, $\mathcal{C} \supseteq \bigcup_{\mathsf{pk}:(\mathsf{pk},\mathsf{sk}) \in \mathsf{KGen}(1^\lambda)} \mathcal{C}_{\mathsf{pk}}$.

Typically, the decryption algorithm $\mathsf{Dec}(\mathsf{sk}, \cdot)$ is only required to be well-defined over $\mathcal{C}_{\mathsf{pk}}$ for $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}(1^\lambda)$, but not necessarily over the entire ciphertext space $\mathcal{C}$ (which includes ciphertexts produced under other public keys). Correspondingly, evaluation correctness is defined upon "valid" ciphertexts.

In this work, we explicitly require the decryption algorithm $\mathsf{Dec}(\cdot, \cdot)$ of an FHE to be *well-defined over the entirety of* $\mathcal{K} \times \mathcal{C}$, in the sense that it always outputs something in the message space $\mathcal{M}$ (albeit the message obtained when decrypting with a wrong key might be unpredictable). We also require the scheme to satisfy a stronger variant of evaluation correctness over all ciphertexts in $\mathcal{C}$. We bundle these extra requirements into *the strong correctness property.*

Correctness. An FHE scheme is correct if the following are satisfied.

– (Decryption Correctness) For any $\lambda \in \mathbb{N}$, any $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}(1^\lambda)$, and any message $m \in \mathcal{M}$, we have that

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m\right] \geq 1 - \mathsf{negl}(\lambda)$$

where the probability is taken over the random coins of Enc.

– (Evaluation Correctness) For any $\lambda \in \mathbb{N}$, any $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}(1^\lambda)$, any positive integer $n \in \mathsf{poly}(\lambda)$, any polynomial-size circuit $\mathsf{C} \in \mathbb{C}_n$, any ciphertexts $(c_1, \ldots, c_n) \in \mathcal{C}_{\mathsf{pk}}^n$, if there exists $m_i = \mathsf{Dec}(\mathsf{sk}, c_i) \in \mathcal{M}$ for all $i \in \{1, \ldots, n\}$, then

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, c) = \mathsf{C}(m_1, \ldots, m_n) : c \leftarrow \mathsf{Eval}(\mathsf{pk}, \mathsf{C}, (c_1, \ldots, c_n))\right] \geq 1 - \mathsf{negl}(\lambda)$$

where the probability is taken over the random coins of Enc and Eval.

The scheme is *perfectly correct* if the above probabilities are exactly 1.

Strong Correctness. A strongly-correct FHE scheme satisfies all below.

– (Decryption Correctness) Same as in the (usual) correctness definition above.
– (Well-Defined Decryption) $\mathsf{Dec}(\cdot, \cdot)$ is well-defined over $\mathcal{K} \times \mathcal{C}$, *i.e.*, for any $(\mathsf{sk}, c) \in \mathcal{K} \times \mathcal{C}$, there exists $m \in \mathcal{M}$ such that $m = \mathsf{Dec}(\mathsf{sk}, c)$.
– (Strong Evaluation Correctness) Evaluation correctness holds even for ciphertexts taken in $\mathcal{C}$. Formally, for any $\lambda \in \mathbb{N}$, any $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}(1^\lambda)$, any positive integer $n \in \mathsf{poly}(\lambda)$, any polynomial-size circuit $\mathsf{C} \in \mathbb{C}_n$, any ciphertexts $(c_1, \ldots, c_n) \in \mathcal{C}^n$ (possibly with $c_i \notin \mathcal{C}_{\mathsf{pk}}$), if there exists $m_i = \mathsf{Dec}(\mathsf{sk}, c_i) \in \mathcal{M}$ for all $i \in \{1, \ldots, n\}$, then

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, c) = \mathsf{C}(m_1, \ldots, m_n) : c \leftarrow \mathsf{Eval}(\mathsf{pk}, \mathsf{C}, (c_1, \ldots, c_n))\right] \geq 1 - \mathsf{negl}(\lambda)$$

where the probability is taken over the random coins of Enc and Eval.

The scheme is *perfectly strongly correct* if the above probabilities are exactly 1.

Security. We recall the standard IND-CPA-security and define a new notion called IK-IND-CPA-security, which combines key privacy and message indistinguishability. We also recall the notion of circular security.

$\mathsf{IND\text{-}CPA}^b_{\mathcal{E},\mathcal{A}}(1^\lambda)$	$\mathsf{IK\text{-}IND\text{-}CPA}^b_{\mathcal{E},\mathcal{A}}(1^\lambda)$	$\mathsf{IND\text{-}CIRC\text{-}CPA}^b_{\mathcal{E},\mathcal{A}}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$	$(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{KGen}(1^\lambda)$	$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
$(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pk})$	$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KGen}(1^\lambda)$	$c_{\mathsf{sk}} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{sk})$
$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$	$(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pk}_0, \mathsf{pk}_1)$	$(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pk})$
$b' \leftarrow \mathcal{A}_2(\mathsf{st}, c)$	$c \leftarrow \mathsf{Enc}(\mathsf{pk}_b, m_b)$	$c_b \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$
return b'	$b' \leftarrow \mathcal{A}_2(\mathsf{st}, c)$	$b' \leftarrow \mathcal{A}_2(\mathsf{st}, c_{\mathsf{sk}}, c_b)$
	return b'	**return** b'

Fig. 2. Security Experiments of FHE (st is the state information of $(\mathcal{A}_1, \mathcal{A}_2)$)

Definition 5 (IND-CPA). *An FHE scheme $\mathcal{E}$ is* IND-CPA*-secure (has indistinguishable messages under chosen-plaintext attack) if for any* PPT *adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, it holds that*

$$\left| \Pr\left[\mathsf{IND\text{-}CPA}^0_{\mathcal{E},\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{IND\text{-}CPA}^1_{\mathcal{E},\mathcal{A}}(1^\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{IND\text{-}CPA}^b_{\mathcal{E},\mathcal{A}}$ is defined in Fig. 2.

Definition 6 (IK-IND-CPA). *An FHE scheme $\mathcal{E}$ is* IK-IND-CPA*-secure (has indistinguishable keys and indistinguishable messages under chosen-plaintext attack) if for any* PPT *adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, it holds that*

$$\left| \Pr\left[\mathsf{IK\text{-}IND\text{-}CPA}^0_{\mathcal{E},\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{IK\text{-}IND\text{-}CPA}^1_{\mathcal{E},\mathcal{A}}(1^\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{IK\text{-}IND\text{-}CPA}^b_{\mathcal{E},\mathcal{A}}$ is defined in Fig. 2.

Definition 7 (Circular Security). *Let $\mathcal{E}$ be an FHE scheme such that $\mathcal{K} = \mathcal{M}$. $\mathcal{E}$ is circular secure if for any* PPT *adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, it holds that*

$$\left| \Pr\left[\mathsf{IND\text{-}CIRC\text{-}CPA}^0_{\mathcal{E},\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{IND\text{-}CIRC\text{-}CPA}^1_{\mathcal{E},\mathcal{A}}(1^\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{IND\text{-}CIRC\text{-}CPA}^b_{\mathcal{E},\mathcal{A}}$ is defined in Fig. 2.

Instantiations. While IND-CPA security is the *de facto* standard of FHE schemes, virtually all of them satisfy the stronger notion of IK-IND-CPA security. This is because FHE ciphertexts are typically indistinguishable from elements uniformly sampled from the ciphertext space (see, *e.g.* [20]).

Typically, FHE schemes are proven to satisfy the standard correctness notion. Below, we show how these schemes can be transformed into one with strong correctness, assuming circular security and the decryption algorithm $\mathsf{Dec}(\cdot,\cdot)$ is well-defined over $\mathcal{K} \times \mathcal{C}$. The former assumption is "for free" as it is already needed for bootstrapping the FHE scheme [18]. The latter is already satisfied by most existing FHE schemes and can be otherwise obtained by artificially extending the decryption algorithm to be well-defined over any input. For the case of FHE schemes based on learning with errors (LWE), $\mathsf{Dec}(\cdot, \cdot)$ typically consists of an inner product of two vectors in $\mathbb{Z}_q^\ell$, followed by rounding. Thus $\mathsf{Dec}(\cdot, \cdot)$ is well defined for *any* pair of vectors in $\mathbb{Z}_q^\ell$ if we set $\mathcal{C} := \mathcal{K} := \mathbb{Z}_q^\ell$.

Let $\mathcal{E}$ be such an FHE scheme. A public key in our transformed scheme $\mathcal{E}'$ is of the form $\mathsf{pk}' = (\mathsf{pk}, c_{\mathsf{sk}})$ where $c_{\mathsf{sk}} = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}, \mathsf{sk})$ is an encryption of the secret key sk under pk. The secret key sk' is identical to sk. The encryption (with input pk) and decryption algorithms of $\mathcal{E}$ and $\mathcal{E}'$ are identical.

The evaluation algorithm $\mathcal{E}'.\mathsf{Eval}$, on input pk', a circuit $\mathsf{C} \in \mathbb{C}_n$, and (not necessarily well-formed) ciphertexts $(c_1, \ldots, c_n) \in \mathcal{C}^n$ works as follows:

– homomorphically decrypts c_i using c_{sk} for each $i \in [n]$, *i.e.*, compute
$$c'_i \leftarrow \mathcal{E}.\mathsf{Eval}(\mathsf{pk}, \mathcal{E}.\mathsf{Dec}(\cdot, c_i), c_{\mathsf{sk}}),$$
– then, evaluates C homomorphically over $(c'_1, \ldots, c'_n)$, *i.e.*, output
$$c' \leftarrow \mathcal{E}.\mathsf{Eval}(\mathsf{pk}, \mathsf{C}, (c'_1, \ldots, c'_n)).$$

Clearly, if $\mathcal{E}$ is IK-IND-CPA-secure and circular secure, then $\mathcal{E}'$ is IK-IND-CPA-secure. To see why $\mathcal{E}'$ has strong correctness, we note that $c_{\mathsf{sk}} \in \mathcal{C}_{\mathsf{pk}}$ by construction and $\mathsf{Dec}(\cdot, c_i)$ is well-defined over $\mathcal{K} = \mathcal{M}$ for all $i \in [n]$ by assumption. Therefore, by the (standard) correctness of $\mathcal{E}$, for all $i \in [n]$, $\mathcal{E}.\mathsf{Dec}(\mathsf{sk}, c'_i) = \mathcal{E}.\mathsf{Dec}(\mathsf{sk}, c_i)$. Next, since $c'_i \in \mathcal{C}_{\mathsf{pk}}$ for all $i \in [n]$, we have $c' \in \mathcal{C}_{\mathsf{pk}}$. Using the (standard) correctness of $\mathcal{E}$ again, if $m_i = \mathcal{E}'.\mathsf{Dec}(\mathsf{sk}', c_i) = \mathcal{E}.\mathsf{Dec}(\mathsf{sk}, c_i)$ for all $i \in [n]$, then $\mathcal{E}'.\mathsf{Dec}(\mathsf{sk}', c') = \mathcal{E}.\mathsf{Dec}(\mathsf{sk}, c') = \mathsf{C}(m_1, \ldots, m_n)$ as we desired.

To draw an analogy to LWE-based schemes, even though $(c_1, \ldots, c_n)$ might have very large noise (with respect to pk), $\mathcal{E}.\mathsf{Eval}$ is executed over c_{sk}, which is well-formed (has small noise) and $(c_1, \ldots, c_n)$ are just constants in the description of the circuits $\mathcal{E}.\mathsf{Dec}(\cdot, c_i)$. This is analogous to Gentry's bootstrapping procedure [18] and works for exactly the same reason.

Our modification essentially introduces an additional bootstrapping step before every homomorphic evaluation. Thus, fast bootstrapping techniques can be applied to make the overhead we added minimal when compared to the cost of the homomorphic evaluation. As the communication complexity of our scheme depends on the rate (message-to-ciphertext size ratio) of the FHE scheme, one can achieve optimal communication (for large enough data blocks) by using the rate-1 FHE [8,19]. It is not hard to see that those schemes also satisfy the notion of IK-IND-CPA security (since ciphertexts are identical to those of [20]).

4.3 Distributed Point Functions

A point function is a function whose images are zero at all points except one.

Definition 8 (Point Function). *A point function $F_{a,b} : \{0,1\}^d \to \{0,1\}^r$, for $a \in \{0,1\}^d$ and $b \in \{0,1\}^r$, is defined by $F_{a,b}(a) = b$, and $F_{a,b}(c) = 0^r$ if $c \neq a$.*

Unless differently specified, we interpret the output domain $\{0,1\}^r$ of F as an Abelian group $\mathbb{G}$ with respect to the group operator $\oplus$.

A distributed point function (DPF) allows secret-sharing a point function f to multiple servers. The servers can locally evaluate the shared function at any point x and produce output shares, which can be combined to recover $f(x)$.

Definition 9 (Distributed Point Function [4]). *For $N \in \mathbb{N}$ and $t \in [N]$, a (t,N)-DPF is a tuple of PPT algorithms $\mathsf{DPF}.(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Dec})$ defined as follows.*

$\underline{\mathsf{DPF}.\mathsf{Gen}(1^\lambda, F_{a,b})}$: *On input the security parameter 1^λ and the description of a point function $F_{a,b}$, the key generation algorithm returns N keys $(k_1, \dots, k_N)$.*

$\underline{\mathsf{DPF}.\mathsf{Eval}(i, k_i, x)}$: *On input a party index i, a key k_i, and a string $x \in \{0,1\}^d$, the evaluation algorithm returns a share s_i.*

$\underline{\mathsf{DPF}.\mathsf{Dec}(s_1, \dots, s_N)}$: *On input a set of shares $(s_1, \dots, s_N)$, the decoding algorithm returns the function output y.*

We consider an N-party additive output decoder for an Abelian group $(\mathbb{G}, +)$ that returns $y = \sum_{i=1}^{N} s_i$ on input $(s_1, \dots, s_N) \in \mathbb{G}^N$. We state a relaxed correctness notion that allows the evaluation algorithm to have an error Δ, and recall the standard notion of security.

Definition 10 (Δ-Correctness). *A (t,N)-$\mathsf{DPF} = (\mathsf{DPF}.\mathsf{Gen}, \mathsf{DPF}.\mathsf{Eval}, \mathsf{DPF}.\mathsf{Dec})$ is correct if there exists an inverse polynomial error bound Δ such that for all $\lambda \in \mathbb{N}$, $x \in \{0,1\}^d$, and point functions $F_{a,b}$,*

$$\Pr\left[\begin{array}{l}(k_1, \dots, k_N) \leftarrow \mathsf{DPF}.\mathsf{Gen}(1^\lambda, F_{a,b}) \\ \{s_i \leftarrow \mathsf{DPF}.\mathsf{Eval}(i, k_i, x)\}_{i \in [N]}\end{array} : \mathsf{DPF}.\mathsf{Dec}(s_1, \dots, s_N) \neq F_{a,b}(x)\right] \leq \Delta.$$

If $\Delta = 0$ then we say that the scheme is perfectly correct.

Definition 11 (Security). *A (t,N)-$\mathsf{DPF} = \mathsf{DPF}.(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Dec})$ is secure if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, subsets $T \subseteq [N]$ such that $|T| = t$, all PPT non-uniform distinguishers $\mathcal{A}$, it holds that*

$$\Pr\left[\begin{array}{l}((a^0, b^0), (a^1, b^1)) \leftarrow \mathcal{A}(1^\lambda) \\ \beta \leftarrow \{0,1\} \\ (k_1, \dots, k_N) \leftarrow \mathsf{DPF}.\mathsf{Gen}(1^\lambda, F_{a^\beta, b^\beta})\end{array} : \beta = \mathcal{A}(\{k_i\}_{i \in T})\right] \leq \mathsf{negl}(\lambda).$$

Boyle *et al.* [4] showed that $(1,2)$-DPF can be built from one-way functions.

Theorem 3 ([4]). *A perfectly-correct $(1,2)$-DPF of $\mathsf{poly}\big(d(\lambda + \log(|\mathbb{G}|))\big)$-size key can be built from one-way functions.*

We also observe that the complexity of the $\mathsf{DPF.Eval}$ algorithm in their construction [4] is dominated by d-many sequential evaluations of a length-doubling PRG. This fact is going to be useful for our later construction.

4.4 Homomorphic Secret Sharing

Homomorphic secret sharing (HSS) can be seen as generalizing a distributed point function where the evaluation algorithm supports the evaluation of more complex circuits. We focus on single-client HSS. In such a scheme, a single client secret shares an input x to multiple servers. These servers can then locally evaluate any circuit C in the supported class of circuits to produce some output shares. The value $C(x)$ can then be recovered by combining these output shares.

Definition 12 (Homomorphic Secret Sharing [6]). *For $N \in \mathbb{N}$, $t \in [N]$, a (t, N)-HSS for a circuit family $\mathcal{C}$ is defined by the following PPT algorithms:*

$\underline{\mathsf{HSS.Gen}(1^\lambda, x)}$: *On input the security parameter 1^λ and an input x, the share generation algorithm returns a set of shares $(s_1, \dots, s_N)$.*

$\underline{\mathsf{HSS.Eval}(i, s_i, C)}$: *On input a party index i, a share s_i, and a circuit $C \in \mathcal{C}$, the evaluation algorithm returns an evaluated share z_i.*

$\underline{\mathsf{HSS.Dec}(z_1, \dots, z_N)}$: *On input a set of shares $(z_1, \dots, z_N)$, the decoding algorithm returns the output y.*

We say that an HSS scheme is *compact* if the size of the output shares does not grow with the size of the circuit given as input to the $\mathsf{HSS.Eval}$ algorithm. We define correctness where the evaluation algorithm may incur an error with probability at most Δ, for some inverse polynomial function Δ.

Definition 13 (Δ-Correctness). *A (t, N)-$\mathsf{HSS} = \mathsf{HSS.(Gen, Eval, Dec)}$ is correct if there exists an inverse polynomial error bound Δ such that for all $\lambda \in \mathbb{N}$, inputs x, and circuits $C \in \mathcal{C}$, we have that*

$$\Pr\left[\begin{array}{l}(s_1, \dots, s_N) \leftarrow \mathsf{HSS.Gen}(1^\lambda, x) \\ \{z_i \leftarrow \mathsf{HSS.Eval}(i, s_i, C)\}_{i \in [N]}\end{array} : \mathsf{HSS.Dec}(z_1, \dots, z_N) \neq C(x)\right] \leq \Delta.$$

Security is defined canonically.

Definition 14 (Security). *A (t, N)-$\mathsf{HSS} = \mathsf{HSS.(Gen, Eval, Dec)}$ is secure if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, subsets $T \subseteq [N]$ such that $|T| = t$, all PPTnon-uniform distinguishers $\mathcal{A}$, it holds that*

$$\Pr\left[\begin{array}{l}(x^0, x^1) \leftarrow \mathcal{A}(1^\lambda) \\ \beta \leftarrow \{0, 1\} \\ (s_1, \dots, s_N) \leftarrow \mathsf{HSS.Gen}(1^\lambda, x^\beta)\end{array} : \beta = \mathcal{A}(\{s_i\}_{i \in T})\right] \leq \mathsf{negl}(\lambda).$$

It is useful to recall a theorem from Boyle *et al.* [5], where they propose an HSS scheme for NC1 circuits assuming the hardness of the DDH problem. There also exists a similar construction based on the hardness of the DCR problem [17].

Theorem 4 ([5]). *If the DDH problem is hard, there exists a compact Δ-correct* $(1, 2)$-HSS *for circuits in NC1, for any inverse polynomial Δ.*

5 Multi-Client ORAM and Its Simulation-based Security

5.1 Syntax

MCORAM was introduced by Maffei *et al.* [29] and later extended to the malicious client setting [30]. Existing MCORAM definitions are mostly verbal, which left many subtleties. We recall (a slightly rephrased version of) its definition.

Definition 15. *An MCORAM scheme for message space $\mathcal{M} \not\supseteq \{\epsilon\}$ consists of a* PPT *algorithm* Setup *and protocols* $(\mathsf{ChMod}, \mathsf{Access})$ *executed between a data owner D, polynomially many independent instances of client C, and a server S:*

$\underline{(\mathsf{pp}, \mathsf{msk}, \bar{M}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, M)}$*: The setup algorithm is run by the database owner D. It inputs the security parameter λ, a size parameter n, and an array $M \in \mathcal{M}^n$ of initial data. It outputs the public parameter* pp *(an implicit input of all other algorithms), the master secret key* msk *(to be kept secret by the owner D), and a database $\bar{M}$ (to be forwarded to the server S).*

$\underline{\langle \epsilon; \mathsf{sk}'; \bar{M}' \rangle \leftarrow \mathsf{ChMod}\langle D(\mathsf{msk}, A_R, A_W); C(\mathsf{sk}, A_R, A_W); S(\bar{M}) \rangle}$*: The data owner D grants access rights to a client C, possibly with the help of the server S, using the change-mode protocol. If C has not joined the system yet, it is assumed that* $\mathsf{sk} = \epsilon$*. The basic model only allows granting additional rights.*

To run ChMod*, D inputs the master (owner) secret key* msk*, a client identifier* $\mathsf{id} \in \{0, 1\}^*$*, and two sets $A_R, A_W \subseteq [n]$ of addresses. C inputs his secret key* sk*, and the same sets of addresses A_R and A_W. The server S inputs the database $\bar{M}$. Supposedly, C will be granted reading rights to A_R, and writing rights to A_W.*

At the end of ChMod*, D outputs a the empty string ϵ. C outputs an updated secret key* sk'*. S outputs a possibly updated database $\bar{M}'$.*

$\underline{\langle m'; \bar{M}' \rangle \leftarrow \mathsf{Access}\langle C(\mathsf{sk}, a, m); S(\bar{M}) \rangle}$*: To access a certain address of the memory, client C engages in the access protocol with the server. The client C inputs its secret key* sk*, an address $a \in [n]$, and some data $m \in \mathcal{M} \cup \{\epsilon\}$. Read access is indicated by $m = \epsilon$. Otherwise, the data $m \neq \epsilon$ is to be written to the address a. The server S inputs $\bar{M}$. Regardless of the type of access, the client outputs some data m' read from the address a, while the server updates its database to $\bar{M}'$.*

It is straightforward to extend the MCORAM syntax and security definitions to the *multi-server setting.* Setup outputs multiple encoded databases

$\bar{M}_1, \ldots, \bar{M}_N$ to be maintained by the respective servers. ChMod becomes an $(N+2)$-party protocol between the database owner D, the client C, and the servers $S_1, \ldots, S_N$. The outputs of D and C remain unchanged, while S_i outputs an updated database $\bar{M}_i'$. Similarly, Access becomes an $(N+1)$-party protocol between the client C and $S_1, \ldots, S_N$. Their outputs are defined analogously.

Although our model allows ChMod and Access to be general multi-party protocols, we are primarily interested in constructions where the servers do not communicate with each other to better justify the non-colluding assumption.

5.2 Correctness and Integrity

An MCORAM scheme should not only be correct but satisfy an even stronger property called integrity (*subsuming correctness*): The database entry at the address a can only be changed by clients having write access to a, other clients who might attempt to maliciously tamper with the data of the honest clients will fail. It is a unique property here and is absent in the single-client setting.

More formally, integrity is modeled by an experiment involving an adversary $\mathcal{A}$. The experiment acts as an honest MCORAM server, provides the interface of an MCORAM instance to $\mathcal{A}$, *i.e.*, $\mathcal{A}$ can request to corrupt a client, request for access permissions on behalf of a client, and access the data. To capture the notion of correctness, $\mathcal{A}$ maintains a plaintext copy of the MCORAM-encoded database, *i.e.*, all accesses are *mirrored to the plaintext copy.* The winning condition of $\mathcal{A}$ is to make the maintained plaintext copy of the database ends up inconsistent with the one encoded in the MCORAM.

Definition 16 (Integrity of MCORAM). *An MCORAM Θ has* integrity *if, for all* PPT *adversaries $\mathcal{A}$, size parameters $n = \mathsf{poly}(\lambda)$, and arrays $M \in \mathcal{M}^n$, with experiment* Int *as defined in Fig. 3, we have*

$$\Pr\left[\mathsf{Int}_{\Theta,\mathcal{A}}(1^\lambda, 1^n, M) = 1\right] \leq \mathsf{negl}(\lambda).$$

Integrity in the multi-server setting is almost identical, except that all oracles now return the views of all servers, which reflects that they are all honest but curious. However, integrity in this setting seems challenging to achieve, especially if we assume that the servers do not communicate with each other. Instead, one may consider a weaker notion known as *accountable integrity* (defined in the single-server setting [29]), which requires that any violation of integrity can be caught after-the-fact. Extending it to the multi-server setting is straightforward.

5.3 Obliviousness

Access in MCORAM is fully specified by (id, a, m), meaning that client id is reading address a (if $m = \epsilon$) or writing m to address a. Obliviousness mandates

```
$\mathsf{Int}_{\Theta,\mathcal{A}}(1^\lambda, 1^n, M)$
$(\mathsf{pp}, \mathsf{msk}, \bar{M}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, M)$
$\mathsf{Corrupt} := \emptyset$
$\mathsf{Read}[a] := \mathsf{Write}[a] := \emptyset \ \forall a \in [n]$
$M' := M$
$\mathbb{O} := \{\mathsf{Corr}\mathcal{O}, \mathsf{ChMod}\mathcal{O}, \mathsf{Access}\mathcal{O}\}$
$(\mathsf{id}^*, a^*) \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pp}, \bar{M}, M)$
$\langle m^*; * \rangle$
    $\leftarrow \mathsf{Access}\langle C(\mathsf{sk}_{\mathsf{id}^*}, a^*, \epsilon); S(\bar{M})\rangle$
$b_0 := (\mathsf{id}^* \in \mathsf{Read}[a^*])$
$b_1 := (\mathsf{Corrupt} \cap \mathsf{Write}[a^*] = \emptyset)$
$b_2 := (M'[a^*] \neq m^*)$
return $b_0 \wedge b_1 \wedge b_2$
```

```
$\mathsf{Corr}\mathcal{O}(\mathsf{id})$
$\mathsf{Corrupt} := \mathsf{Corrupt} \cup \{\mathsf{id}\}$
return $\mathsf{sk}_{\mathsf{id}}$
```

```
$\mathsf{ChMod}\mathcal{O}(\mathsf{id}, A_R, A_W, C^*)$
for $a \in A_R$ do $\mathsf{Read}[a] := \mathsf{Read}[a] \cup \{\mathsf{id}\}$
for $a \in A_W$ do $\mathsf{Write}[a] := \mathsf{Write}[a] \cup \{\mathsf{id}\}$
if $\mathsf{id} \in \mathsf{Corrupt}$ then
    $\langle \epsilon; (*; \mathsf{view}_{C^*}); (\bar{M}; \mathsf{view}_S)\rangle$
        $\leftarrow \mathsf{ChMod}\langle D(\mathsf{msk}, A_R, A_W); C^*; S(\bar{M})\rangle$
else
    $\langle \epsilon; \mathsf{sk}_{\mathsf{id}}; (\bar{M}; \mathsf{view}_S)\rangle$
        $\leftarrow \mathsf{ChMod}\langle D(\mathsf{msk}, A_R, A_W);$
                $C(\mathsf{sk}_{\mathsf{id}}, A_R, A_W); S(\bar{M})\rangle$
return $(\mathsf{view}_{C^*}, \mathsf{view}_S)$ // $\mathsf{view}_{C^*}$ can be $\epsilon$
```

```
$\mathsf{Access}\mathcal{O}(\mathsf{id}, a, m, C^*)$
if $\mathsf{id} \in \mathsf{Write}[a] \wedge m \neq \epsilon$ then $M'[a] := m$
if $\mathsf{id} \in \mathsf{Corrupt}$ then
    $\langle (*; \mathsf{view}_{C^*}); (\bar{M}; \mathsf{view}_S)\rangle$
        $\leftarrow \mathsf{Access}\langle C^*; S(\bar{M})\rangle$
else
    $\langle *; (\bar{M}; \mathsf{view}_S)\rangle$
        $\leftarrow \mathsf{Access}\langle C(\mathsf{sk}_{\mathsf{id}}, a, m); S(\bar{M})\rangle$
return $(\mathsf{view}_{C^*}, \mathsf{view}_S)$ // $\mathsf{view}_{C^*}$ can be $\epsilon$
```

Fig. 3. MCORAM's Integrity against Malicious Clients and Honest-but-Curious Server

that such information would not be leaked to any other parties, unless the access is write access and the parties have read access to a.

More formally, (*indistinguishability-based*) obliviousness is modeled by a pair of experiments, labeled by $b = 0, 1$ respectively, involving an adversary $\mathcal{A}$. As in the integrity experiment, the experiments provide the interface of an instance of MCORAM to $\mathcal{A}$, with some differences. First, $\mathcal{A}$ has to provide malicious server codes to the interfaces, which models the setting where the server is always trying to compromise clients' obliviousness. Second, the interface for the access protocol is parameterized by the bit b (which specifies the experiment) and takes as input *two* access instructions $(\mathsf{id}_\beta, a_\beta, m_\beta)$ for $\beta \in \{0, 1\}$. The interface would execute instruction labeled with $\beta = b$. After some interactions with the interface, $\mathcal{A}$ would output a bit b', which can be interpreted as a guess of b. An MCORAM is said to be (indistinguishably) oblivious against malicious clients if the probabilities of $\mathcal{A}$ outputting 1 in either experiment are negligibly close.

```
$\mathsf{Obl}^b_{\Theta,\mathcal{A}}(1^\lambda, 1^n, M)$

$(\mathsf{pp}, \mathsf{msk}, \bar{M}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, M)$
$\mathsf{Corrupt} := \mathsf{ChAddr} := \emptyset$
$\mathsf{Read}[a] := \emptyset\ \forall a \in [n]$
$\mathbb{O} := \{\mathsf{Corr}\mathcal{O}, \mathsf{ChMod}\mathcal{O}, \mathsf{Access}\mathcal{O}_b\}$
$b' \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pp}, \bar{M}, M)$
**return** $b'$

$\mathsf{Corr}\mathcal{O}(\mathsf{id})$

**ensure** $\forall\ a \in \mathsf{ChAddr},\ \mathsf{id} \notin \mathsf{Read}[a]$
$\mathsf{Corrupt} := \mathsf{Corrupt} \cup \{\mathsf{id}\}$
**return** $\mathsf{sk}_{\mathsf{id}}$

$\mathsf{ChMod}\mathcal{O}(\mathsf{id}, A_R, A_W, C^*, S^*)$

**for** $a \in A_R$ **do** $\mathsf{Read}[a] := \mathsf{Read}[a] \cup \{\mathsf{id}\}$
**if** $\mathsf{id} \in \mathsf{Corrupt}$ **then**
  **ensure** $\mathsf{ChAddr} \cap A_R = \emptyset$
  $\langle \epsilon; (*; \mathsf{view}_{C^*}); (*; \mathsf{view}_{S^*})\rangle$
     $\leftarrow \mathsf{ChMod}\langle D(\mathsf{msk}, A_R, A_W); C^*; S^*\rangle$
**else**
  $\langle \epsilon; \mathsf{sk}_{\mathsf{id}}; (*; \mathsf{view}_{S^*})\rangle$
     $\leftarrow \mathsf{ChMod}\langle D(\mathsf{msk}, A_R, A_W);$
              $C(\mathsf{sk}_{\mathsf{id}}, A_R, A_W); S^*\rangle$
**return** $(\mathsf{view}_{C^*}, \mathsf{view}_S)$ // $\mathsf{view}_{C^*}$ can be $\epsilon$

$\mathsf{Access}\mathcal{O}_b((\mathsf{id}_0, a_0, m_0), (\mathsf{id}_1, a_1, m_1), S^*)$

$\beta_0 := \big(\mathsf{Corrupt} \cap \{\mathsf{id}_0, \mathsf{id}_1\} = \emptyset\big)$
$\beta_1 := \big((a_0, m_0) \neq (a_1, m_1)\big) \wedge (m_0 \neq \epsilon)$
$\beta_1 := \beta_1 \wedge (\mathsf{Read}[a_0] \cap \mathsf{Corrupt} \neq \emptyset)$
$\beta_2 := \big((a_0, m_0) \neq (a_1, m_1)\big) \wedge (m_1 \neq \epsilon)$
$\beta_2 := \beta_2 \wedge (\mathsf{Read}[a_1] \cap \mathsf{Corrupt} \neq \emptyset)$
**if** $\beta_0 \vee \beta_1 \vee \beta_2$ **then** **return** $\perp$
$\mathsf{ChAddr} := \mathsf{ChAddr} \cup \{a_0, a_1\}$
$\langle *; (*; \mathsf{view}_{S^*})\rangle \leftarrow \mathsf{Access}\langle C(\mathsf{sk}_{\mathsf{id}_b}, a_b, m_b); S^*\rangle$
**return** $\mathsf{view}_{S^*}$
```

Fig. 4. Obliviousness Experiment of MCORAM against Malicious Clients and Server

Definition 17 (Indistinguishability-based Obliviousness). *An MCORAM scheme Θ is indistinguishably oblivious against malicious clients if, for all* $\mathsf{PPT}\mathcal{A}$*, all λ and $n = \mathsf{poly}(\lambda)$, all arrays $M \in \mathcal{M}^n$, with* Obl *as in Fig. 4,*

$$\left|\Pr\left[\mathsf{Obl}^0_{\Theta,\mathcal{A}}(1^\lambda, 1^n, M) = 1\right] - \Pr\left[\mathsf{Obl}^1_{\Theta,\mathcal{A}}(1^\lambda, 1^n, M) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

So far, we followed Maffei *et al.* [29] and defined an indistinguishability-based obliviousness definition. However, when constructing higher-level protocols, it is often more convenient to prove security based on simulation-based security notions of the building blocks. We thus propose a new simulation-based obliviousness definition for MCORAM, which turns out to be an equivalent one.

Our *simulation-based obliviousness* notion is also modeled by a pair of experiments involving an adversary $\mathcal{A}$, called the real and ideal experiment, respectively. Both experiments provide the interface of an MCORAM instance to $\mathcal{A}$. However, the way that queries to the interface are answered varies greatly.

```
$\mathsf{Real\text{-}Obl}_{\Theta,\mathcal{A}}(1^\lambda, 1^n, M)$

$(\mathsf{pp}, \mathsf{msk}, \bar{M}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, M)$
$\mathsf{Corrupt} := \mathsf{ChAddr} := \emptyset$
$\mathsf{Read}[a] := \emptyset \ \forall a \in [n]$
$\mathbb{O} := \{\mathsf{Corr}\mathcal{O}, \mathsf{ChMod}\mathcal{O}, \mathsf{Access}\mathcal{O}\}$
$b' \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pp}, \bar{M}, M)$
return $b'$

$\mathsf{Corr}\mathcal{O}(\mathsf{id})$

ensure $\forall\, a \in \mathsf{ChAddr},\ \mathsf{id} \notin \mathsf{Read}[a]$
$\mathsf{Corrupt} := \mathsf{Corrupt} \cup \{\mathsf{id}\}$
return $\mathsf{sk}_{\mathsf{id}}$

$\mathsf{ChMod}\mathcal{O}(\mathsf{id}, A_R, A_W, C^*, S^*)$

for $a \in A_R$ do $\mathsf{Read}[a] := \mathsf{Read}[a] \cup \{\mathsf{id}\}$
if $\mathsf{id} \in \mathsf{Corrupt}$ then
    ensure $\mathsf{ChAddr} \cap A_R = \emptyset$
    $\langle \epsilon; (*; \mathsf{view}_{C^*}); (*; \mathsf{view}_{S^*}) \rangle$
        $\leftarrow \mathsf{ChMod}\langle D(\mathsf{msk}, A_R, A_W); C^*; S^* \rangle$
else
    $\langle \epsilon; \mathsf{sk}_{\mathsf{id}}; (*; \mathsf{view}_{S^*}) \rangle$
        $\leftarrow \mathsf{ChMod}\langle D(\mathsf{msk}, A_R, A_W);$
                    $C(\mathsf{sk}_{\mathsf{id}}, A_R, A_W); S^* \rangle$
return $(\mathsf{view}_{C^*}, \mathsf{view}_S)$ // $\mathsf{view}_{C^*}$ can be $\epsilon$

$\mathsf{Access}\mathcal{O}(\mathsf{id} \notin \mathsf{Corrupt}, a, m, S^*)$

if $\mathsf{Read}[a] \cap \mathsf{Corrupt} = \emptyset \wedge m \neq \epsilon$ then
    $\mathsf{ChAddr} := \mathsf{ChAddr} \cup \{a\}$
$\langle *; (*; \mathsf{view}_{S^*}) \rangle \leftarrow \mathsf{Access}\langle C(\mathsf{sk}_{\mathsf{id}}, a, m); S^* \rangle$
return $\mathsf{view}_{S^*}$
```

Fig. 5. Real Experiment for Obliviousness against Malicious Clients and Server

In the real experiment, the interface is backed by a real execution of the MCORAM instance (as in the integrity experiment), where $\mathcal{A}$ needs to provide the malicious server code (as in the indistinguishability-based obliviousness experiment). Answering a query in the ideal experiment generally invokes a simulator $\mathcal{S}$ with the leakage of the query as the input. For example, upon receiving a query (id, a, m) to the interface for the access protocol, if $\mathcal{A}$ has read access to a and $m \neq \epsilon$, then $\mathcal{S}$ is given (a, m). Otherwise, $\mathcal{S}$ is given no information (other than the fact that the query is issued to the access interface). In any case, given such a leakage, $\mathcal{S}$ is supposed to simulate the response of a real execution. After some interactions, $\mathcal{A}$ would output a bit b', which can be interpreted as a guess of whether it has interacted with the real experiment or the ideal experiment. An MCORAM is said to be *semantically oblivious* against malicious clients if the probabilities of $\mathcal{A}$ outputting 1 in either experiment are negligibly close.

Definition 18 (Semantic Obliviousness). *An MCORAM scheme Θ is semantically oblivious against malicious clients if, for all* PPT *adversaries $\mathcal{A}$, all λ and $n = \mathsf{poly}(\lambda)$, and all arrays $M \in \mathcal{M}^n$, there exists a* PPT *simulator $\mathcal{S}$, with* Real-Obl *and* Ideal-Obl *as defined in Figs. 5 and 6 respectively, such that*

$$\left| \Pr\left[\mathsf{Real\text{-}Obl}_{\Theta,\mathcal{A}}(1^\lambda, 1^n, M) = 1 \right] - \Pr\left[\mathsf{Ideal\text{-}Obl}_{\Theta,\mathcal{A},\mathcal{S}}(1^\lambda, 1^n, M) = 1 \right] \right| \leq \mathsf{negl}(\lambda).$$

```
$\mathsf{Ideal\text{-}Obl}_{\Theta,\mathcal{A},\mathcal{S}}(1^\lambda, 1^n, M)$
$(\mathsf{pp}, \mathsf{td}, \bar{M}) \leftarrow \mathcal{S}(\text{`Setup'}, 1^\lambda, 1^n, M)$
$\mathsf{Corrupt} := \mathsf{ChAddr} := \emptyset$
$\mathbb{O} := \{\mathsf{Corr}\mathcal{O}, \mathsf{ChMod}\mathcal{O}, \mathsf{Access}\mathcal{O}\}$
$\mathsf{Read} :=$ Empty dictionaries
$b' \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pp}, \bar{M}, M)$
return $b'$

$\mathsf{Corr}\mathcal{O}(\mathsf{id})$
ensure $\forall\ a \in \mathsf{ChAddr},\ \mathsf{id} \notin \mathsf{Read}[a]$
$\mathsf{Corrupt} := \mathsf{Corrupt} \cup \{\mathsf{id}\}$
$(\mathsf{sk}_{\mathsf{id}}, \mathsf{td}) \leftarrow \mathcal{S}(\text{`Corrupt'}, \mathsf{td}, \mathsf{id})$
return $\mathsf{sk}_{\mathsf{id}}$

$\mathsf{ChMod}\mathcal{O}(\mathsf{id}, A_R, A_W, C^*, S^*)$
for $a \in A_R$ do $\mathsf{Read}[a] := \mathsf{Read}[a] \cup \{\mathsf{id}\}$
if $\mathsf{id} \in \mathsf{Corrupt}$ then
  ensure $\mathsf{ChAddr} \cap A_R = \emptyset$
  $(\mathsf{view}_{C^*}, \mathsf{view}_{S^*}, \mathsf{td})$
      $\leftarrow \mathcal{S}^{C^*,S^*}(\text{`CorrChMod'}, \mathsf{td}, \mathsf{id}, A_R, A_W)$
  return $(\mathsf{view}_{C^*}, \mathsf{view}_{S^*})$
else
  $(\mathsf{view}_{S^*}, \mathsf{td}) \leftarrow \mathcal{S}^{S^*}(\text{`ChMod'}, \mathsf{td}, \mathsf{id}, A_R, A_W)$
return $(\mathsf{view}_{C^*}, \mathsf{view}_{S})$ // $\mathsf{view}_{C^*}$ can be $\epsilon$

$\mathsf{Access}\mathcal{O}(\mathsf{id} \notin \mathsf{Corrupt}, a, m, S^*)$
if $\mathsf{Read}[a] \cap \mathsf{CorrAddr} \neq \emptyset \wedge m \neq \epsilon$ then
  $(\mathsf{view}_{S^*}, \mathsf{td}) \leftarrow \mathcal{S}^{S^*}(\text{`CorrAccess'}, \mathsf{td}, a, m)$
else
  if $m \neq \epsilon$ then $\mathsf{ChAddr} := \mathsf{ChAddr} \cup \{a\}$
  $(\mathsf{view}_{S^*}, \mathsf{td}) \leftarrow \mathcal{S}^{S^*}(\text{`Access'}, \mathsf{td})$
return $\mathsf{view}_{S^*}$
```

Fig. 6. Ideal Experiment for Obliviousness against Malicious Clients and Server

The above two definitions can be shown equivalent using arguments for proving similar statements in encryption. See the full version for formal treatment.

Extending obliviousness to the multi-server setting where at most t of the N servers are corrupt is slightly more complicated. To model this, we modify the security experiments such that all N servers are initially honest, and at most t of them can be corrupted using a modified $\mathsf{Corr}\mathcal{O}$ oracle. Correspondingly, the inputs of the modified $\mathsf{ChMod}\mathcal{O}$ oracle and $\mathsf{Access}\mathcal{O}$ oracle now include at most t pieces of malicious codes $\{S_j^*\}$ for the respective servers, such that S_j^* will be used to generate the communication transcript if server j is corrupt.

6 FHE-based Single-Server Construction

6.1 Formal Description

Fix a database size $n \in \mathbb{N}$ with $n = \mathsf{poly}(\lambda)$. Let cPRF be a constrained PRF family (Section 4.1) with domain $[n] \cup \{0\}$. Let $\mathcal{E}.(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be an FHE scheme (Section 4.2) with message space $\mathcal{M}_{\mathcal{E}}$. Let $\Sigma.(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Verify})$ be a signature scheme with message space $\mathcal{M}_{\Sigma} := \{0,1\}^\lambda \times \mathcal{M}_{\mathcal{E}}$. For any array $\bar{M}$,

```
$\mathsf{Setup}(1^\lambda, 1^n, M = (m_1, \ldots, m_n))$

$\mathsf{msk} \leftarrow \mathsf{cPRF.KGen}(1^\lambda)$
$(\mathsf{pk}_0^{\mathcal{E}}, \mathsf{sk}_0^{\mathcal{E}}) := \mathcal{E}.\mathsf{KGen}(1^\lambda)$
for $k \in [n]$ do
    $r_{R,k} := \mathsf{cPRF.Eval}(\mathsf{msk}, (0, k))$
    $r_{W,k} := \mathsf{cPRF.Eval}(\mathsf{msk}, (1, k))$
    $(\mathsf{pk}_k^{\mathcal{E}}, \mathsf{sk}_k^{\mathcal{E}}) := \mathcal{E}.\mathsf{KGen}(1^\lambda; r_{R,k})$
    $(\mathsf{pk}_k^{\Sigma}, \mathsf{sk}_k^{\Sigma}) := \Sigma.\mathsf{KGen}(1^\lambda; r_{W,k})$
    $\bar{M}[k] := \bar{m}_k \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_k^{\mathcal{E}}, m_k)$
$\mathsf{pp} := \left\{\mathsf{pk}_0^{\mathcal{E}}, (\mathsf{pk}_k^{\mathcal{E}}, \mathsf{pk}_k^{\Sigma})\right\}_{k=1}^{n}$
return $(\mathsf{pp}, \mathsf{msk}, \bar{M})$

$\mathsf{ChMod}\langle D, C, S\rangle$

$D(\mathsf{msk}, A_R, A_W)$
    ensure $A_R, A_W \subseteq [n]$
    $X := (\{0\} \times A_R) \cup (\{1\} \times A_W)$
    $\mathsf{sk}' := \mathsf{cPRF.Constrain}(\mathsf{msk}, X)$
    send $K$ to $C$
    return $\epsilon$
$C(\mathsf{sk}, A_R, A_W)$
    receive $\mathsf{sk}'$ from $D$
    return $\mathsf{sk}'$
$S(\bar{M})$
    return $\bar{M}' := \bar{M}$

$\mathsf{Access}\langle C, S\rangle$

$C(\mathsf{sk}_{\mathsf{id}}, a, m)$
    receive $r$ from $S$
    $r_{R,a} := \mathsf{cPRF.cEval}(\mathsf{sk}, (0, a))$
    $r_{W,a} := \mathsf{cPRF.cEval}(\mathsf{sk}, (1, a))$
    $(\mathsf{pk}_a^{\mathcal{E}}, \mathsf{sk}_a^{\mathcal{E}}) := \mathcal{E}.\mathsf{KGen}(1^\lambda; r_{R,a})$
    $(\mathsf{pk}_a^{\Sigma}, \mathsf{sk}_a^{\Sigma}) := \Sigma.\mathsf{KGen}(1^\lambda; r_{W,k})$
    $(\tilde{\mathsf{pk}}^{\mathcal{E}}, \tilde{\mathsf{sk}}^{\mathcal{E}}) \leftarrow \mathcal{E}.\mathsf{KGen}(1^\lambda)$
    $\sigma \leftarrow \Sigma.\mathsf{Sig}(\mathsf{sk}_a^{\Sigma}, (r, m))$
    $\tilde{c}_0 \leftarrow \mathcal{E}.\mathsf{Enc}(\tilde{\mathsf{pk}}^{\mathcal{E}}, a)$
    if $m \neq \epsilon$ then
        $c_1 \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_a^{\mathcal{E}}, \sigma), c_2 \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_a^{\mathcal{E}}, m)$
    else
        $c_1 \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_0^{\mathcal{E}}, 0), c_2 \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_0^{\mathcal{E}}, 0)$
    send $(\tilde{\mathsf{pk}}^{\mathcal{E}}, \tilde{c}_0, c_1, c_2)$ to $S$
    receive $\tilde{c}_0'$ from $S$
    $\bar{m}' \leftarrow \mathcal{E}.\mathsf{Dec}(\tilde{\mathsf{sk}}^{\mathcal{E}}, \tilde{c}_0')$
    return $m' \leftarrow \mathcal{E}.\mathsf{Dec}(\mathsf{sk}_a^{\mathcal{E}}, \bar{m}')$
$S(\bar{M})$
    send $r \leftarrow_\$ \{0,1\}^\lambda$ to $C$
    receive $(\tilde{\mathsf{pk}}^{\mathcal{E}}, \tilde{c}_0, c_1, c_2)$ from $C$
    send $\tilde{c}_0' \leftarrow \mathcal{E}.\mathsf{Eval}(\tilde{\mathsf{pk}}^{\mathcal{E}}, \mathsf{Read}_{\bar{M}}, \tilde{c}_0)$ to $C$
    for $k \in [n]$ do
        $\bar{m}_k' \leftarrow \mathcal{E}.\mathsf{Eval}(\mathsf{pk}_k^{\mathcal{E}}, \mathsf{Write}_{\mathsf{pk}_k^{\Sigma}, r}, (c_1, c_2, \bar{m}_k))$
        $\bar{M}'[k] := \bar{m}_k'$
    return $\bar{M}'$
```

Fig. 7. FHE-based Single-Server MCORAM Construction

nonce $r \in \{0,1\}^\lambda$, and public key pk of Σ, we define the following circuits:

$$\mathsf{Read}_{\bar{M}}(\mathsf{addr}) := \bar{M}[\mathsf{addr}]$$

$$\mathsf{Write}_{\mathsf{pk},r}(\mathsf{sig}, \mathsf{data}', \mathsf{data}) := \begin{cases} \mathsf{data}' & \text{if } \Sigma.\mathsf{Verify}(\mathsf{pk}, (r, \mathsf{data}'), \mathsf{sig}) = 1 \\ \mathsf{data} & \text{otherwise} \end{cases}$$

With the above, Fig. 7 presents an MCORAM Θ for the message space $\mathcal{M}_{\mathcal{E}}$. We highlight some key steps. We assume for now that the data owner D generates the keys $(\mathsf{pk}_k^{\mathcal{E}}, \mathsf{sk}_k^{\mathcal{E}})$ and $(\mathsf{pk}_k^{\Sigma}, \mathsf{sk}_k^{\Sigma})$ for $k \in [n] \cup \{0\}$ during setup, and publishes all public keys as public parameters. Naturally, the keys indexed by $k \in [n]$

corresponds to the n addresses of the database, while the keys indexed by 0 are reserved for other purposes. The database at the server S is $\bar{M} = \{\bar{m}_k\}_{k\in[n]}$, where $\bar{m}_k = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_k^{\mathcal{E}}, m_k)$. Reading and writing rights to an address $a \in [n]$ is granted to a client C by simply sending to C the key $\mathsf{sk}_a^{\mathcal{E}}$ and sk_a^{Σ}, respectively.

To obliviously access an address $a \in [n]$, the client C first requests a nonce r from the server S. C then generates a fresh FHE key $\tilde{\mathsf{pk}}^{\mathcal{E}}$, and uses it to encrypt a in c_0. Then, for a write access, C uses sk_a^{Σ} to sign r and the data m to be stored, and encrypts the resulting signature σ in c_1 and m in c_2. For a read operation, C sets both σ and m to 0, and uses $\mathsf{pk}_0^{\mathcal{E}}$ to generate c_1 and c_2 instead.

As S is supposedly oblivious to the address, it homomorphically evaluates the reading circuit $\mathsf{Read}_{\bar{M}}$ parameterized by the entire database $\bar{M}$ over c_0. This results in a ciphertext encrypting m_a under $\tilde{\mathsf{pk}}^{\mathcal{E}}$, whose secret key is only known by C. S also evaluates the writing circuit $\mathsf{Write}_{\mathsf{pk}_k^{\Sigma},r}$ over $(c_1, c_2, \bar{m}_k)$ for each address $k \in [n]$. Under the hood of FHE, $\mathsf{Write}_{\mathsf{pk}_k^{\Sigma},r}$ checks if σ is a valid signature of (r, m) w.r.t. pk_k^{Σ}, and if so (C has writing rights to k and intends to write m there), returns a ciphertext $\bar{m}'_k$ encrypting the new m under $\mathsf{pk}_k^{\mathcal{E}}$. If not, $\bar{m}'_k$ would be encrypting m_k (the original data) under $\mathsf{pk}_k^{\mathcal{E}}$. Regardless of the result (which S is oblivious to), S updates the k-th entry of the database to $\bar{m}'_k$.

To reduce the size of the master secret key, D can derive the $\mathcal{E}$ and Σ keys using the pseudorandomness generated by cPRF. Correspondingly, D sends the appropriately constrained PRF keys, so that the clients can re-derive the $\mathcal{E}$ and Σ secret keys. If cPRF features succinct constrained keys (of size sublinear in the description size of the constraining set), then the MCORAM features succinct client keys (of size sublinear in the number of permitted addresses).

6.2 Security

Integrity requires that data written honestly can be successfully read by honest clients, which largely follows from the correctness of the building blocks. The more challenging requirement is to ensure the adversary can not overwrite entries without write access. We first use the correctness of FHE and the signature scheme to argue that, unless a valid signature of a random nonce is given, an entry would never be overwritten, then we argue for its unforgeability.

Obliviousness is intuitive, too, because a client always sends a fresh public key and three FHE ciphertexts during access, regardless of the access type. Although the ciphertexts are generated using keys that may depend on the access, we can rely on the key privacy of FHE and argue that they are still indistinguishable. The proofs for our theorems can be found in the full version.

Theorem 5. *If* cPRF *is selective-input correct and pseudorandom,* $\mathcal{E}$ *is strongly correct,* Σ *is correct, and* Σ *is EUF-CMA-secure, then* Θ *has integrity.*

Theorem 6. *If* cPRF *is selective-input pseudorandom, and* $\mathcal{E}$ *is IND-CPA-secure and IK-IND-CPA-secure, then* Θ *is oblivious.*

6.3 Access Rights Revocation

Generic techniques for revocation are compatible with our construction. First of all, the data owner could always re-encrypt database entries and/or re-generate the corresponding signature verification keys. However, this requires the data owner to re-grant the access rights of the refreshed entries from scratch. Using a constrained PRF for a powerful enough class of constraints, we can save the data owner from some troubles in always re-granting the keys to the clients.

Recall that cPRF is used for deriving the address-dependent secret keys. To support revocation, we consider an equivalent formulation of cPRF, where the PRF key is constrained by a predicate P, such that the constrained key allows evaluations of the PRF over the inputs x satisfying $P(x) = 1$. The core idea is to put the latest client revocation list as an input of P for deriving the latest keys.

In more detail, the data owner publishes (*e.g.*, via the server) the client revocation lists $\mathcal{L}_{\text{Read},a}$ and $\mathcal{L}_{\text{Write},a}$ for each $a \in [n]$, which contain the identifiers of clients whose read access and respectively write access to address a have been revoked. Suppose client id is entitled to read access to addresses $a \in A_R$ and write access to addresses $a \in A_W$, respectively. The data owner delegates to client id a PRF key constrained with respect to the following predicate $P_{A_R,A_W,\mathsf{id}}$ parameterized by A_R, A_W, and id (*i.e.*, they are embedded in the constraint key):

$P_{A_R,A_W,\mathsf{id}}(\mathsf{op}, \mathsf{addr}, \mathsf{CRL})$: If $(\mathsf{id} \notin \mathsf{CRL}) \wedge ((\mathsf{op} = \text{Read} \wedge \mathsf{addr} \in A_R) \vee (\mathsf{op} = \text{Write} \wedge \mathsf{addr} \in A_W))$, return 1; else return 0.

The last input CRL can be changing ($\mathcal{L}_{\text{Read}}$ or $\mathcal{L}_{\text{Write}}$). We still use the PRF output to generate the signing and verification keys for the address a, but it would be the PRF value on $(\text{Write}, a, \mathcal{L}_{\text{Write},a})$ for write access, for example. If $a \in A_W$ and $\mathsf{id} \notin \mathcal{L}_{\text{Write},a}$, client id can evaluate the PRF on $(\text{Write}, a, \mathcal{L}_{\text{Write},a})$, and hence derive the signing key needed for write access to a.

To revoke (more) clients their write access to a, the data owner informs the server of a new verification key (which is a PRF value of the new blacklist). Read access can be revoked similarly, except that the data owner would need to re-encrypt those database entries whose revocation policies have been changed.

7 DPF-based Multi-Server Construction

7.1 Our Distributed Point Function

Let $(\mathsf{DPF}'.\mathsf{Gen}, \mathsf{DPF}'.\mathsf{Eval}, \mathsf{DPF}'.\mathsf{Dec})$ be a $(1, 2)$-DPF such that $\mathsf{DPF}'.\mathsf{Eval}$ is in NC1, and let $(1, 2)$-$\mathsf{HSS} = (\mathsf{HSS}.\mathsf{Gen}, \mathsf{HSS}.\mathsf{Eval}, \mathsf{HSS}.\mathsf{Dec})$ be a homomorphic secret sharing as defined in Section 4.4 for NC1 circuits. Fig. 8 shows our $(3, 4)$-DPF

```
DPF.Gen($1^\lambda, F_{a,b}$)
    $(k_1, k_2) \leftarrow$ DPF'.Gen($1^\lambda, F_{a,b}$)
    $(s_1, s_2) \leftarrow$ HSS.Gen($1^\lambda, k_1$)
    $(s_3, s_4) \leftarrow$ HSS.Gen($1^\lambda, k_2$)
    return $(s_1, s_2, s_3, s_4)$

DPF.Eval($i, k_i, x$)
    if $i \in \{1, 3\}$ then $j := 1$
    if $i \in \{2, 4\}$ then $j := 2$
    $\ell := 2$
    if $i \leq 2$ then $\ell := 1$
    $C \leftarrow$ DPF'.Eval($\ell, \cdot, a$)
    $z \leftarrow$ HSS.Eval($j, k_i, C$)
    return $z$

DPF.Dec($z_1, z_2, z_3, z_4$)
    $y_1 \leftarrow$ HSS.Dec($z_1, z_2$)
    $y_2 \leftarrow$ HSS.Dec($z_3, z_4$)
    $z \leftarrow$ DPF'.Dec($y_1, y_2$)
    return $z$
```

Fig. 8. $(3, 4)$-DPF Construction

construction. The $(2, 3)$-DPF follows a straightforward modification, which we include in the full version.

Theorem 7 (Correctness). *Let $\hat{\Delta}$ and $\tilde{\Delta}$ be inverse polynomials. Let $(1, 2)$-DPF′ be a $\hat{\Delta}$-correct distributed point function and let $(1, 2)$-HSS be a $\tilde{\Delta}$-correct homomorphic secret sharing. Our construction in Fig. 8 is a Δ-correct $(3, 4)$-DPF, for some inverse polynomial Δ.*

Theorem 8 (Security). *Let $(1, 2)$-DPF′ be a secure distributed point function and let $(1, 2)$-HSS be a secure homomorphic secret sharing. The construction in Fig. 8 is a secure $(3, 4)$-DPF.*

The proofs of Theorems 7 and 8 can be found in the full version.

Instantiations. By Theorem 4, there exists a $(1, 2)$-HSS for NC1 circuits with share size $\mathsf{poly}\left(\lambda|a|\right)$, which is Δ-correct for any inverse polynomial Δ, assuming the hardness of the DDH (or DCR) problem. By Theorem 3, there exists a perfectly-correct $(1, 2)$-DPF′ with $\mathsf{poly}\left(d(\lambda + \log(|\mathbb{G}|))\right)$ key size, where $\{0, 1\}^d$ is the domain of the point function, assuming the existence of one-way functions. What is left to be shown is that our $(3, 4)$-DPF is efficient when plugging in these two building blocks. More precisely, we will show that DPF′.Eval of the $(1, 2)$-DPF′ is computable by an NC1 circuit. Recall that the complexity of the algorithm of [4] is dominated by d calls for a length-doubling PRG. For a point function with a polynomial-size domain, we can set $d = c\log(\lambda)$, for some constant c, then implementing the length-doubling PRG with a construction in NC0 (such as Goldreich PRG [23]) gives us an evaluation algorithm computable by an NC1 circuit. The size of the resulting keys of our $(3, 4)$-DPF is $\mathsf{poly}\left(\lambda d \cdot (\lambda + \log(|\mathbb{G}|))\right) = \mathsf{poly}\left(\lambda c \log(\lambda) \cdot (\lambda + \log(|\mathbb{G}|))\right)$. We thus obtain:

Corollary 1. *If the DDH or DCR problem is hard and there exists a PRG in NC0, there exists a Δ-correct $(3, 4)$-DPF for any inverse polynomial Δ, for functions with polynomial-size domain $\{0, 1\}^d$ with key size $\mathsf{poly}\left(\lambda d \cdot (\lambda + \log(|\mathbb{G}|))\right)$.*

```
$\underline{\mathsf{Setup}(1^\lambda, 1^n, 1^N, M = (m_1, \ldots, m_n))}$
for $k \in [n]$ do
  $(\mathsf{pk}_k^{\mathcal{E}}, \mathsf{sk}_k^{\mathcal{E}}) \leftarrow \mathcal{E}.\mathsf{KGen}(1^\lambda)$
  $\bar{m}'_k \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_k^{\mathcal{E}}, m_k)$
$\bar{M}' := (\bar{m}'_1, \ldots, \bar{m}'_n)$
for $k \in [N]$
    sample $\bar{M}_k$ s.t. $\bar{M}' = \sum_{i \in [N]} \bar{M}_i$
for $(i, j) \in [N]^2$ do $\bar{M}_{i,j} := \bar{M}_i$
$\mathsf{pp} := (\mathsf{pk}_1^{\mathcal{E}}, \ldots, \mathsf{pk}_n^{\mathcal{E}})$
$\mathsf{msk} := (\mathsf{sk}_1^{\mathcal{E}}, \ldots, \mathsf{sk}_n^{\mathcal{E}})$
return $(\mathsf{pp}, \mathsf{msk}, \bar{M}_{1,1}, \ldots, \bar{M}_{N,N})$

$\underline{\mathsf{Join}(\mathsf{msk}, \mathsf{id})}$
return $\mathsf{sk}_{\mathsf{id}} := \epsilon$

$\underline{\mathsf{ChMod}\langle D, C_{\mathsf{id}}, (S_{1,1}, \ldots, S_{N,N})\rangle}$
$\underline{D(\mathsf{msk}, \mathsf{id}, A \subseteq [n])}$
  $\tilde{\mathsf{sk}}_{\mathsf{id}} := \emptyset$
  for $a \in A$ do
    $\tilde{\mathsf{sk}}_{\mathsf{id}} := \tilde{\mathsf{sk}}_{\mathsf{id}} \cup \{(a, \mathsf{sk}_a^{\mathcal{E}})\}$
  send $\tilde{\mathsf{sk}}_{\mathsf{id}}$ to $C_{\mathsf{id}}$
  return $\epsilon$
$\underline{C_{\mathsf{id}}(\mathsf{sk}_{\mathsf{id}}, A \subseteq [n])}$
  receive $\tilde{\mathsf{sk}}_{\mathsf{id}}$
  return $\mathsf{sk}'_{\mathsf{id}} := \mathsf{sk}_{\mathsf{id}} \cup \tilde{\mathsf{sk}}_{\mathsf{id}}$
$\underline{S_{i,j}(\bar{M}_{i,j}), (i, j) \in [N]^2}$
  return $\bar{M}'_{i,j} := \bar{M}_{i,j}$

$\underline{\mathsf{Access}\langle C_{\mathsf{id}}, (S_{1,1}, \ldots, S_{N,N})\rangle}$
$\underline{C_{\mathsf{id}}(\mathsf{sk}_{\mathsf{id}}, a, m)}$
  $(k_1, \ldots, k_N) \leftarrow \mathsf{DPF.Gen}(1^\lambda, F_{a,1})$
  for $(i, j) \in [N]^2$ send $k_j$ to $S_{i,j}$
$\underline{S_{i,j}(\bar{M}_{i,j}), (i, j) \in [N]^2}$
  receive $k_j$
  for $a' \in [n]$ do
    $z_{a'} \leftarrow \mathsf{DPF.Eval}(j, k_j, a')$
  send $c_{i,j} = \sum_{a' \in [n]} \bar{M}_{i,j}[a'] \cdot z_{a'}$ to $C_{\mathsf{id}}$
$\underline{C_{\mathsf{id}}(\mathsf{sk}_{\mathsf{id}}, a, m)}$
  receive $(c_{1,1}, \ldots, c_{N,N})$
  $\bar{m}' := \sum_{(i,j) \in [N]^2} c_{i,j}$
  if $(x, \mathsf{sk}_a^{\mathcal{E}}) \in \mathsf{sk}_{\mathsf{id}}$ then
    $m' \leftarrow \mathcal{E}.\mathsf{Dec}(\mathsf{sk}_a^{\mathcal{E}}, \bar{m}')$
  if $m = \epsilon$ then $b := 0$
  else $\bar{m} \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_a^{\mathcal{E}}, m), b := \bar{m} - \bar{m}'$
  $(k_1, \ldots, k_N) \leftarrow \mathsf{DPF.Gen}(1^\lambda, F_{a,b})$
  for $(i, j) \in [N]^2$ send $k_i$ to $S_{i,j}$
  return $m'$
$\underline{S_{i,j}(\bar{M}_{i,j}), (i, j) \in [N]^2}$
  receive $k_i$
  for $a' \in [n]$ do
    $z_{a'} \leftarrow \mathsf{DPF.Eval}(i, k_i, a')$
    $\bar{M}'_{i,j}[a'] := \bar{M}_{i,j}[a'] + z_{a'}$
  return $\bar{M}'_{i,j}$
```

Fig. 9. DPF-based Multi-Server MCORAM Construction

7.2 Multi-Client ORAM from Distributed Point Functions

As described in the introduction, there exists a folklore way (*e.g.*, [9]) to construct distributed ORAM (DORAM) with stateless client from any (t, N)-DPF with linear reconstruction. Such a DORAM can be further transformed into a multi-server MCORAM by equipping it with access control. Incorporating reading rights (while achieving obliviousness) is straightforward via encryption. Granting meaningful writing rights and achieving integrity, however, seems impossible in the setting where the servers cannot communicate (even indirectly).

To show the legitimacy of an update, the client needs to prove the knowledge of a witness for a statement about the database, which is secret-shared among the servers. As any single server has no information about the statement, the proof cannot be verified during the access. (We discuss an alternative later). In Fig. 9, we propose a transformation in a simplified setting where all clients have writing rights to all addresses by default, so the ChMod protocol is used only for granting reading rights (via decryption keys). In this setting, the syntax of ChMod can be simplified, which inputs a set of addresses A (*cf.*, A_R and A_W).

Fig. 9 assumes N^2 servers, indexed by $(i, j) \in [N]^2$, with at most t of them collude. Each entry $M[a]$ of the initial data array M is first encrypted with an independent public key pk_a of a public-key encryption scheme $\mathcal{E}$, and then secret-shared using the additive N-out-of-N secret sharing scheme. The (i, j)-th server gets the i-th share $\bar{M}_i$ of the ciphertext. We use independent encryption and decryption keys for each address for simplicity. The master and client secret key sizes can be reduced using constrained PRFs as in the FHE-based construction.

To access address a, the client generates fresh DPF keys $(k_1, \ldots, k_N)$ for the point function $F_{a,1}$, and sends k_j to server (i, j) for $(i, j) \in [N]^2$. Using the additive reconstruction property of the DPF, the (i, j)-th server can compute the j-th share of $\bar{M}_i[a] = \sum_{a' \in [n]} F_{a,1}(a')\bar{M}_i[a]$. Collecting all N^2 shares, the client can recover $\bar{M}[a]$, and decrypt it using sk_a to get $M[a]$. Regardless of whether logical access is a read or a write, the client must write something to ensure obliviousness. In case of a write access, the client encrypts the new data item as $\bar{m}$, and sets $b := \bar{m} - \bar{M}[a]$; in case of a read access, the client sets $b := 0$. The client then generates another fresh tuple of DPF keys $(k_1, \ldots, k_N)$ for the point function $F_{a,b}$, and sends k_i to server (i, j) for $(i, j) \in [N]^2$. Using the reconstruction property again, the i-th row of servers can obtain, for each $a' \in [n]$, the same i-th share of $\bar{M}'[a']$ being $\bar{M}[a']$ for $a' \neq a$, $\bar{m}$ otherwise.

Properties of The Resulting Multi-Server MCORAM. Since the DPF is resilient against the disclosure of any $t < N$ shares, the multi-server (MC)ORAM scheme is secure against a t/N^2 fraction of corruptions. One can show that the scheme is oblivious with a simple reduction to the security of DPF.

Meaningful selective writing rights can be granted by settling for accountable integrity. The techniques for it are rather standard [29]. Roughly, assuming there is an underlying versioning system (as in a typical storage system) that stores each (encrypted) update instruction, we additionally require the client to anonymously sign the update with traceable signatures (*e.g.*, [15]). The data owner can then trace the misbehaving party via the anonymity revocation mechanism.

Consistency is another issue. Due to the underlying HSS [5], our DPF might fail with a certain probability. This is undesirable, especially for write operations, as it would leave the database in a corrupted state. Fortunately, the same HSS [5] allows the servers to detect potential errors; thus, they can abort accordingly.

Finally, note that we can even use different DPF algorithms of different parameters for read and write. This allows some tunable trade-offs, *e.g.*, using a (t, N)-DPF for write and a (t', N')-DPF for read brings the threshold/server-ratio to $\min\{t, t'\}/(NN')$. Specifically, using a (t, N)-DPF for write and a $(1, 1)$-DPF (*i.e.*, a PIR scheme) for read brings the threshold/server-ratio down to t/N.

8 Concluding Remarks

Since many techniques for constructing single-client ORAM break down completely when the client can be corrupt, it was unclear whether the poly-logarithmic communication complexity of ORAM can be attained by MCORAM with an access control mechanism and obliviousness against fellow clients. We devise a cross-key trial evaluation technique and two new distributed point functions for building (multi-server) MCORAM with poly-logarithmic communication complexity. Besides, existing MCORAM definitions are indistinguishability based and may not be readily applicable in higher cryptographic applications. This paper also filled in this gap. Our study benefits the applications of MCORAM for building higher cryptographic primitives and enriches our understanding of homomorphic secret sharing. Application-wise, our MCORAM is especially useful for private anonymous data access in an outsourced setting.

References

1. Blass, E., Mayberry, T., Noubir, G.: Multi-client oblivious RAM secure against malicious servers. In: ACNS. pp. 686–707. Springer (2017)
2. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Asiacrypt Part II. pp. 280–300. Springer (2013)
3. Boyle, E., Chung, K., Pass, R.: Oblivious parallel RAM and applications. In: TCC-A Part II. pp. 175–204. Springer (2016)
4. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing. In: Eurocrypt Part II. pp. 337–367. Springer (2015)
5. Boyle, E., Gilboa, N., Ishai, Y.: Breaking the circuit size barrier for secure computation under DDH. In: Crypto Part I. pp. 509–539. Springer (2016)
6. Boyle, E., Gilboa, N., Ishai, Y., Lin, H., Tessaro, S.: Foundations of homomorphic secret sharing. In: ITCS. pp. 21:1–21:21 (2018)
7. Boyle, E., Ishai, Y., Pass, R., Wootters, M.: Can we access a database both locally and privately? In: TCC part II. pp. 662–693. Springer (2017)
8. Brakerski, Z., Döttling, N., Garg, S., Malavolta, G.: Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In: TCC Part II. pp. 407–437. Springer (2019)
9. Bunn, P., Katz, J., Kushilevitz, E., Ostrovsky, R.: Efficient 3-party distributed ORAM. In: SCN. pp. 215–232. Springer (2020)
10. Camenisch, J., Dubovitskaya, M., Neven, G.: Oblivious transfer with access control. In: CCS. pp. 131–140 (2009)
11. Canetti, R., Holmgren, J., Richelson, S.: Towards doubly efficient private information retrieval. In: TCC Part II. pp. 694–726. Springer (2017)

12. Canetti, R., Lombardi, A., Wichs, D.: Non-interactive zero knowledge and correlation intractability from circular-secure FHE. Cryptology ePrint Archive, Report 2018/1248 (2018)
13. Chen, B., Lin, H., Tessaro, S.: Oblivious parallel RAM: improved efficiency and generic constructions. In: TCC-A, Part II. pp. 205–234. Springer (2016)
14. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. J. ACM **45**(6), 965–981 (1998)
15. Chow, S.S.M.: Real traceable signatures. In: SAC. pp. 92–107. Springer (2009)
16. Doerner, J., abhi shelat: Scaling ORAM for secure computation. In: CCS. pp. 523–535 (2017)
17. Fazio, N., Gennaro, R., Jafarikhah, T., III, W.E.S.: Homomorphic secret sharing from Paillier encryption. In: ProvSec. pp. 381–399. Springer (2017)
18. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University (2009)
19. Gentry, C., Halevi, S.: Compressible FHE with applications to PIR. In: TCC Part II. pp. 438–464. Springer (2019)
20. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Crypto Part I. pp. 75–92. Springer (2013)
21. Gilboa, N., Ishai, Y.: Distributed point functions and their applications. In: Eurocrypt. pp. 640–658. Springer (2014)
22. Goldreich, O.: Towards a theory of software protection and simulation by oblivious RAMs. In: STOC. pp. 182–194 (1987)
23. Goldreich, O.: A primer on pseudorandom generators, vol. 55. American Mathematical Soc. (2010)
24. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM **33**(4), 792–807 (1986)
25. Goodrich, M.T., Mitzenmacher, M., Ohrimenko, O., Tamassia, R.: Privacy-preserving group data access via stateless oblivious RAM simulation. In: SODA. pp. 157–167 (2012)
26. Hamlin, A., Ostrovsky, R., Weiss, M., Wichs, D.: Private anonymous data access. In: Eurocrypt Part II. pp. 244–273. Springer (2019)
27. Hoang, T., Ozkaptan, C.D., Yavuz, A.A., Guajardo, J., Nguyen, T.: S^{3}ORAM: A computation-efficient and constant client bandwidth blowup ORAM with Shamir secret sharing. In: CCS. pp. 491–505 (2017)
28. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: Single database, computationally-private information retrieval. In: FOCS. pp. 364–373 (1997)
29. Maffei, M., Malavolta, G., Reinert, M., Schröder, D.: Privacy and access control for outsourced personal records. In: S&P. pp. 341–358 (2015)
30. Maffei, M., Malavolta, G., Reinert, M., Schröder, D.: Maliciously secure multi-client ORAM. In: ACNS. pp. 645–664. Springer (2017)
31. Sahin, C., Zakhary, V., Abbadi, A.E., Lin, H., Tessaro, S.: TaoStore: Overcoming asynchronicity in oblivious data storage. In: S&P. pp. 198–217 (2016)
32. Wang, F., Yun, C., Goldwasser, S., Vaikuntanathan, V., Zaharia, M.: Splinter: Practical private queries on public data. In: NSDI. pp. 299–313 (2017)