Private Laconic Oblivious Transfer with Preprocessing

Rishabh Bhadauria[1], Nico Döttling[2], Carmit Hazay[3], and Chuanwei Lin[2]

[1] Georgetown University
rishabh.bhadauria@georgetown.edu
[2] CISPA Helmholtz Center for Information Security
nico.doettling@gmail.com, chuanwei.lin@cispa.de
[3] Bar-Ilan University
carmit.hazay@biu.ac.il

Abstract. Laconic cryptography studies two-message protocols that securely compute on large amounts of data with minimal communication cost. Laconic oblivious transfer (OT) is a central primitive where the receiver's input is a large database DB and the sender's inputs are two messages m_0, m_1 along with an index i, such that the receiver learns the message determined by the choice bit DB_i. OT becomes even more useful for secure computation when considering its laconic variants, which offer succinctness and round optimality. However, existing constructions are not practically efficient because they rely on heavy cryptographic machinery and non-black-box techniques.

In this work, we initiate the study of *laconic OT correlations*, where the model allows an offline phase to generate the correlations later used in a lightweight online phase. Our correlation is conceptually simple, captured by an inner product computation, and enables us to achieve a *private* laconic OT protocol where the sender's index i is also hidden from the receiver. Our construction is the first private laconic OT with database-dependent preprocessing based solely on symmetric-key assumptions, achieving sublinear online computational complexity for the receiver. Furthermore, we enhance our construction with updatability and receiver privacy. Finally, we demonstrate the applications of private laconic OT to laconic function evaluation for RAM programs and laconic private set intersection with preprocessing.

1 Introduction

Secure two-party computation protocols allow two parties to jointly compute a function over their inputs while ensuring that no more information is revealed than what can be inferred from the output itself. A central primitive in this area is *oblivious transfer* (OT) [Rab81, Kil88a] where, the receiver holds a bit b, and the sender holds two messages m_0 and m_1. At the end of the protocol, the receiver learns the message m_b, but nothing about m_{1-b}. Conversely, the sender learns nothing about the bit b. The importance of oblivious transfer stems from the fact that it is *complete* for secure two-party computation [Kil88b], in the sense that any efficient two-party function $F(x, y)$ can be securely computed given access to a secure OT protocol.

While early results in secure two-party computation established the feasibility of this task [Yao82, Rab81, Kil88a, GMW87], the resulting protocols were prohibitively expensive in several respects. In particular, these protocols incurred large overheads in terms of round, communication, and computational complexity compared to the underlying insecure protocol task.

To address some of these challenges, the *offline/online paradigm* introduces a preprocessing phase that generates correlated randomness and shifts the expensive computations to the offline phase, thereby reducing the burden during the online phase. Originating from the work of Beaver [Bea92], this setting has been extensively explored for secure multi-party computation. A substantial body of works has been studying different types of two-party correlations [IKNP03, Bea96, BDOZ11, DPSZ12, NNOB12, IKM+13, DKL+13, DZ13, DNNR17, HOSS18, CDE+18, BGI19, Cou19, BCG+19a, BCG+19b, YWZ20, YWL+20, BLN+21, BCG+22, BCG+23] including multiplication triples, one-time truth-tables, garbled circuit correlations, and OT and OLE correlations. Protocols designed under such a paradigm offer significant efficiency gains and are conceptually simpler. Once the preprocessing is finished, the online phase only involves lightweight computations, typically symmetric key operations or combinatorial operations.

Orthogonally, the advent of fully homomorphic encryption (FHE) [Gen09, BV11b, BV11a, GSW13] had introduced a new approach to secure two-party computation, enabling protocols in just two

messages and with asymptotically optimal communication. In addition, FHE allows the receiver to delegate almost the entire computational burden to the sender. In particular, this allows for protocols where the receiver has a very large input, while the communication grows with this size. This drawback makes FHE-based approaches unsuitable for scenarios where the sender, often a resource-constrained client, cannot handle such a computational load or communication overhead.

An emerging trend termed *laconic cryptography* addresses the delegation imbalance described above. It studies two-message protocols where the receiver has a large input x, while the sender has a small input y. The key requirement for *laconism* is that the communication size be independent of, or at least sublinear in, the size of x. In this setting, the computational burden shifts to the receiver rather than the sender, leading to what is often referred to as *reverse delegation*.

Just as oblivious transfer is a fundamental primitive in standard two-party computation protocols, laconic oblivious transfer [CDG$^+$17] plays a similarly central role in laconic cryptographic protocols. In a laconic OT protocol, the receiver commits to a large database $\mathsf{DB} \in \{0.1\}^n$ via a succinct hash value. Given this hash value, the sender, whose inputs are two messages m_0, m_1 and an index $i \in [n]$, computes a ciphertext c from which the receiver can decrypt m_{DB_i}. For sender security, we require that the receiver learn nothing about $m_{1-\mathsf{DB}_i}$. In terms of communication efficiency, the size of the hash value is independent of the size of the database.

Generalizing the concept of laconic OT, laconic function evaluation (LFE), introduced by Quach et al. [QWW18], allows any function to be inversely delegated to the receiver. In an LFE protocol, the receiver computes a succinct hash value of a large circuit C. Given this hash value, the sender can encrypt his input y into a ciphertext c. Using the circuit C, the receiver can decrypt c to $\mathsf{C}(y)$. Regarding security, we require that the receiver learn nothing about y beyond $\mathsf{C}(y)$. When two parties want to compute a function $F(x, y)$ in this setting, the receiver will hash a circuit $\mathsf{C}[x](\cdot)$ that computes the function $F(x, \cdot)$ with her input x hardwired. Regarding assumptions and communication efficiency (or laconism), Quach et al. [QWW18] constructed an LFE scheme for circuits based on the learning with error (LWE) assumption [Reg05], and the communication size scales with the depth of the circuit C but not with its overall size.

Private laconic OT [DGI$^+$19] imposes a stricter security requirement than laconic OT, where the index i is hidden from the receiver and is useful for more advanced applications such as laconic function evaluation for RAM [DHMW24]. In this setting, the receiver commits to a database DB via a succinct hash value. The sender, whose input is a RAM program P of runtime T, encrypts the program P into a ciphertext c. With the database DB in hand, the receiver can decrypt c to P^{DB}, i.e., the output of P when executed on DB. For communication efficiency, the size of the hash h is independent of the size of the database DB, whereas the ciphertext c scales linearly with the runtime T. Regarding security, the receiver must learn nothing about P beyond its output P^{DB} and the runtime T. Private laconic OT guarantees full security by hiding the index i from the receiver, thereby concealing the access pattern of P^{DB} on the database.

These works mentioned above represent early results on laconic OT and related concepts, achieving near-optimal asymptotic communication and computational overheads. However, they rely heavily on non-black-box use of cryptographic primitives, requiring intensive public-key operations within garbled circuits. As a result, they are primarily considered feasibility results with limited practical implications.

More recently, several works have made significant progress towards truly efficient laconic cryptography by avoiding using garbled circuits. Döttling et al. [DKL$^+$23] introduced a construction from the LWE assumption, Green et al. [GJL23] proposed one from a new assumption over pairing-friendly elliptic curves in the generic group model, and Fleischhacker et al. [FHAS24] developed a construction in the algebraic group model. However, these constructions fall short when applied to advanced applications of laconic cryptography, such as RAM-LFE, because they still rely on non-black-box use of the underlying laconic OT protocols. This reliance negates much of the potential efficient gains even with the improved laconic OT, as cryptographic operations within the circuits introduce significant computational overhead. To fully unlock the benefits of laconic OT in advanced applications, we need lightweight constructions where, even if laconic OT is integrated within a circuit, the operations are limited to lightweight tasks such as arithmetic computations and bitwise operations as opposed to costly public-key operations. Additionally, from a theoretical perspective, exploring constructions based on weaker assumptions could lead to more efficient and versatile protocols. This would make them easier to implement and expand the applicability of laconic OT to a broader range of advanced cryptographic systems.

In summary, although laconic cryptography harbors great potential for advanced applications beyond theoretical interest, practical implementations remain elusive due to the prohibitive costs by the non-black-box techniques in current constructions. To achieve a conceptually simple and practically efficient solution for advanced applications, merely optimizing existing techniques is insufficient. Instead, a fundamental rethinking of the core laconic primitives and a new model tailored to the laconic setting will be required.

1.1 Our Results

Inspired by the extensive research on OT correlations, we initiate the study of *laconic OT correlations* and introduce preprocessing into laconic cryptography. Our contributions focus on constructing *private laconic OT with preprocessing*, drawing remote inspiration from the recent advances in private information retrieval (PIR) with preprocessing [CK20]. We present the following key results:

Our Model. We propose a relaxed notion of *laconism* by modeling laconic OT with preprocessing in a two-server offline/online paradigm. In our model, two non-colluding servers hold replicas of the database. During the offline phase, the offline server generates *database-dependent correlations* and becomes silent after the preprocessing. In the online phase, the client, acting as the OT sender, sends a single message to the online server, which acts as the OT receiver. Regarding *laconism*, we ensure that the size of the correlations is sublinear in the size of the database. Furthermore, the entire protocol consists of only two messages: one from the offline server to the client in the offline phase and another from the client to the online server in the online phase. The key advantage of our new model is that it allows us to decouple the expensive public-key operations from simpler computations by abstracting the particular randomness correlations required for the underlying protocol, shifting the costly computation to the offline phase, and leaving the online phase to handle only lightweight primitives.

Our model can be reduced to the standard two-party setting by having the sender and receiver emulate the role of the offline server, e.g., using laconic function evaluation [QWW18], while maintaining sublinear communication in the overall protocol. Notably, due to the modular nature of our model, any generic or custom protocol can replace the offline server in a black-box manner. Given that the primary goal of this work is to identify the structured correlations that are essential for laconic OT, we leave the challenge of developing simplified, customized protocols that can effectively replace the offline server for future study.

Our Construction. We process to introduce the first laconic OT correlation that can be captured by *inner product computations*. Like standard OT correlations, these laconic OT correlations are single-use. The use of these correlations allows for the construction of a private, laconic OT protocol with database-dependent preprocessing based solely on symmetric cryptographic primitives. In our model, the offline server generates these correlations locally and subsequently transmits them to the sender for use during the online phase. Inner product computations have demonstrated their versatility across various domains, serving as fundamental components in areas such as zero-knowledge proof systems, such as [BCC$^+$16]. We therefore anticipate that insights derived from prior work may prove beneficial in the present context, particularly with regard to enhancing security against malicious adversaries and unlocking further optimization opportunities. By drawing on techniques from these diverse areas, the scope of inner product-based correlations can be expanded and new possibilities.

With the inner product-based laconic OT correlations, we present the first *private* laconic OT with preprocessing which achieves sublinear online computational complexity for the receiver in the two-server setting. Specifically, assuming the existence of one-way functions, for a database of n bits and a security parameter λ, our scheme has the following complexities,

- Offline communication: $\tilde{O}_\lambda(\sqrt{n})$,
- Online communication: $\tilde{O}_\lambda(\sqrt{n})$,
- Computational overhead for the sender during the online phase: $O_\lambda(\sqrt{n})$,
- Computational overhead for the receiver during the online phase: $O_\lambda(\sqrt{n})$,

where $\tilde{O}(\cdot)$ hides arbitrary polylogarithmic factors, and $O_\lambda(\cdot)$ hides arbitrary polynomial factors in the security parameter λ.

Previous work on private laconic OT includes the construction by [DGI+19], which only achieved ciphertexts of size sublinear in n and incurred linear complexity for the receiver. More recently, the private laconic OT implied by the work of Dong et al. [DHMW24] achieved fully compact ciphertexts that achieve polylogarithmic complexity for the receiver by leveraging the recent breakthrough of Lin et al. [LMW23], who constructed a doubly efficient private information retrieval scheme under standard assumptions. Our construction, operating in the alternate two-server model, achieves ciphertexts of size sublinear in n and sublinear complexity for the receiver, relying exclusively on symmetric primitives.

We also show how to lift our construction to achieve updatability and receiver privacy, ensuring a broader class of applicability.

Applications. Our private laconic OT with preprocessing has broad potential across various applications, such as laconic function evaluation for RAM programs (RAM-LFE) and laconic private set intersection (PSI). It offers a more efficient alternative to standard protocols, especially in scenarios that align with the two-server model. Specifically, by incorporating preprocessing, query latency is reduced by shifting the heavier computations to the offline phase, making it an ideal solution for time-sensitive operations.

RAM-LFE. In our two-server model, for a RAM program with runtime T and a database of size n, our RAM-LFE with preprocessing has the following complexities,

- Offline communication: $\tilde{O}_\lambda(T \cdot \sqrt{n})$,
- Online communication: $\tilde{O}_\lambda(T \cdot \sqrt{n})$,
- Computational overhead for the sender during the online phase: $O_\lambda(T \cdot \sqrt{n})$,
- Computational overhead for the receiver during the online phase: $O_\lambda(T \cdot \sqrt{n})$.

Previous constructions [DHMW24, DHM+24] suffer from the bottleneck of evaluating public-key primitives within garbled circuits, which is notoriously expensive. Operating in the two-server model, our RAM-LFE with preprocessing still uses garbled circuits, but only for symmetric primitives. Given the extensive research on evaluating symmetric primitives with garbled circuits, e.g., [PAP24], our approach reduces computational overhead and also results in smaller circuits, thereby improving concrete communication efficiency.

Laconic PSI. We further show that given a collision-free hash function for a receiver with a database of size n (derived from a large dataset) and a sender with a dataset of size m, our laconic PSI with preprocessing has the following complexities,

- Offline communication: $\tilde{O}_\lambda(\sqrt{n})$,
- Online communication: $\tilde{O}_\lambda(m \cdot \sqrt{n})$,
- Computational overhead for the sender during the online phase: $O_\lambda(m \cdot \sqrt{n})$,
- Computational overhead for the receiver during the online phase $O_\lambda(m \cdot \sqrt{n})$.

While previous constructions [ABD+21, ALOS22, DKL+23] have the server time scaling linearly in the large input size, our laconic PSI with preprocessing, operating in a different two-server model, achieves sublinear online time.

Future Directions. Our work opens a new avenue for private laconic OT with preprocessing, drawing inspiration from advances in the PIR domain. However, due to the unique requirements of laconism and the reversed roles of sender and receiver, PIR techniques cannot be directly applied to our context. Nevertheless, some follow-up research in PIR may be relevant to our future work, such as the exploration of efficiency lower bounds and the development of methods for generating reusable correlations on the sender side. These provide fertile ground for further research on private laconic OT with preprocessing.

Furthermore, recalling that our correlations are based on inner product computations, it remains open to explore simpler custom-made protocols that compute these correlations, and whether insights from other fields, such as inner product arguments in zero-knowledge proofs, could enhance security and lead to further optimizations, potentially enabling a private laconic OT with preprocessing that provides malicious security. Another interesting direction is to explore alternative types of laconic OT correlations.

2 Technical Overview

To introduce our protocol, we start by revisiting the concept of a *laconic OT* scheme and then relax the notion of laconism to accommodate *preprocessing* for generating the *laconic OT correlations* in an offline/online two server model. We then propose our laconic OT correlations, which can be captured by inner product computation, and show a laconic OT protocol with weak *index privacy*. To strengthen the privacy guarantee, we explore the potential of adopting techniques such as bias sampling and parallel repetition from recent work on private information retrieval (PIR) with preprocessing [CK20]. However, the differing functionality, where the roles of the sender and the receiver are reversed, and the unique requirements of laconism, which restrict the protocol to only two messages, make it insufficient to simply incorporate these techniques. To overcome these challenges, we integrate secret sharing alongside these two techniques and construct the first *private* laconic OT with preprocessing. Lastly, we demonstrate how the protocol can be extended to support *updatability* and *receiver privacy*, along with its applications to *laconic function evaluation for RAM programs* and *laconic private set intersection*.

2.1 Our New Model for Laconic OT

Laconic OT. We begin by outlining the overview of a laconic OT protocol. In such a protocol [CDG$^+$17], two parties are involved: the receiver holding a database $\mathsf{DB} \in \{0,1\}^n$ where each entry DB_i serves as a choice bit, and the sender holding an index $i \in [n]$ and a pair of messages m_0, m_1. At the end of the protocol, the receiver learns the message m_{DB_i} but nothing about the other message. The protocol consists of four algorithms:

- A *setup algorithm* that takes the security parameters and outputs a public parameter pp.
- A *hash algorithm* which takes pp and a database $\mathsf{DB} \in \{0,1\}^n$ as input and outputs a digest digest and a state st.
- A *sender algorithm* which takes pp, a digest digest, a index $i \in [n]$ and two messages m_0, m_1 as input and produces a ciphertext e.
- A *receiver algorithm* which takes st, a index i and a ciphertext e as input and outputs a message m.

Correctness requires that, given the digest of the database DB and a ciphertext computed against it, a valid database index i, and two messages m_0, m_1, the receiver can decrypt the ciphertext to m_{DB_i}. Security ensures that a semi-honest receiver does not learn anything about $m_{1-\mathsf{DB}_i}$. It should be noted that in a laconic OT protocol, in contrast to standard oblivious transfer, there is no receiver privacy, i.e., no security guarantee against a corrupted sender. Efficiency, or *laconism*, requires that the size of digest is independent of the database size n and that the runtime of both the sender and the receiver algorithm are at most polynomially dependent on $\log n$.

Two-server Model. Recall that current laconic OT constructions either rely on heavy cryptographic machinery or are not suited for advanced cryptographic applications that require non-black-box use of laconic OT. To address this, we introduce preprocessing into laconic cryptography to achieve a conceptually simpler and more efficient solution. The offline/online paradigm, commonly used in MPC, allows computationally expensive tasks to be shifted to an offline phase, leaving a lightweight online phase. By leveraging this approach, we generate database-dependent correlations during the offline phase, which are then used in the online phase to reduce the computational overhead.

To enable database-dependent preprocessing, we model our protocol within a two-server offline/online setting involving three parties: a client, an offline server, and an online server. Both the offline and online servers possess a replica of the database DB, but crucially, they do not interact or collude, as in the models from [BGKW88, CK20]. In this setting, the client acts as the OT sender, and the online server acts as the OT receiver. In the offline phase, the offline server locally generates laconic OT correlations and sends them to the client. Then, in the online phase, the client (OT sender) sends a single message to the online server (OT receiver), completing the laconic OT protocol.

Regarding laconism, we require that the size of the correlations produced during preprocessing be sublinear in the database size. Additionally, this model involves only two messages: the first one from the offline server to the client (the OT sender) and the second one from the client (OT sender) to the online server (OT receiver). This model offers a relaxed version compared to the standard two-message laconic cryptographic protocols, where a short digest, generated by a hash algorithm, is in size independent of or at least sublinear in that of the database.

2.2 Our (Private) Laconic OT Construction

Laconic OT Correlations. We observe that the OT correlations can be realized via inner product computations. The computation is conceptually simple: for two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^n$ and all $i \in [n]$, it holds that

$$\langle \mathbf{a}, \mathbf{b} \rangle = a_i \cdot b_i + \langle \mathbf{a}_{-i}, \mathbf{b} \rangle$$

where $\mathbf{a}_{-i}$ represents the vector $\mathbf{a}$ with its i-th entry a_i set to 0.

Recall that in the standard laconic OT functionality, the receiver holds a database DB, while the sender holds an index i and a pair of messages, m_0 and m_1. For now, we assume the sender also holds the correlation $(\mathbf{v}, \langle \mathbf{v}, \mathsf{DB} \rangle) \in \mathbb{F}_q^n \times \mathbb{F}_q$ such that $v_i \neq 0$. The inner product computation shown above can then be reformulated as

$$\langle \mathbf{v}, \mathsf{DB} \rangle - v_i \cdot \mathsf{DB}_i = \langle \mathbf{v}_{-i}, \mathsf{DB} \rangle$$

where $\mathbf{v}_{-i}$ is the vector $\mathbf{v}$ with the i-th element set to 0, denoted as a punctured vector in this work.

Based on this formula, if the sender sends the punctured vector $\mathbf{v}_{-i}$ to the receiver, the receiver, with access to the database DB, can compute $\langle \mathbf{v}_{-i}, \mathsf{DB} \rangle$, thus learning either $\langle \mathbf{v}, \mathsf{DB} \rangle$ if $\mathsf{DB}_i = 0$ or $\langle \mathbf{v}, \mathsf{DB} \rangle - v_i$ if $\mathsf{DB}_i = 1$. Since v_i is uniform from the receiver's perspective, she cannot infer the other value. Leveraging this observation, these two values can serve as keys: the sender can encrypt m_0 with $\langle \mathbf{v}, \mathsf{DB} \rangle$ and m_1 with $\langle \mathbf{v}, \mathsf{DB} \rangle - v_i$, ensuring that the receiver learns only m_{DB_i} and nothing about the other message.

To instantiate this idea, we require an IND-CPA secure symmetric encryption scheme. Specifically, the sender sets two distinct keys as $k_0 = \langle \mathbf{v}, \mathsf{DB} \rangle$ and $k_1 = \langle \mathbf{v}, \mathsf{DB} \rangle - v_i$. The ciphertexts are then encrypted as $\mathsf{ctxt}_b = \mathsf{Enc}_{k_b}(m_b \| 0^\lambda)$ for $b \in \{0, 1\}$ where 0^λ is a padding to ensure the ciphertexts are of the same size and indistinguishable under encryption. Afterward, the sender sends the punctured vector $\mathbf{v}_{-i}$ and the two ciphertexts ctxt_0 and ctxt_1 to the receiver.

Upon receiving this message, the receiver computes the key $k = \langle \mathbf{v}_{-i}, \mathsf{DB} \rangle$. Then it decrypts both ciphertexts using this key to recover m_{DB_i}: The decryption yields the message concatenated with the padding, but only the one corresponding to m_{DB_i} will have a valid padding structure of the form $m \| 0^\lambda$. Since the receiver can only learn the key depending on her choice bit DB_i, the receiver cannot learn the other message $m_{1-\mathsf{DB}_i}$.

Index Privacy. For the advanced applications we aim to achieve, standard laconic OT is insufficient. Instead, we require *private* laconic OT with a stronger privacy guarantee by hiding the index i from the receiver. In the strawman protocol outlined above, we may seem to achieve such a notion since the receiver does not explicitly get the index i. However, the transmitted vector $\mathbf{v}_{-i}$, zeroed out at the sender's index i, could leak information about i due to zero entries at specific positions. To address this, we need to obfuscate the punctured vector $\mathbf{v}_{-i}$, ensuring that it does not reveal the exact position of the sender's index. To implement this, we rely on techniques from private information retrieval (PIR) with preprocessing [CK20]. Specifically, we construct sparse vectors $\mathbf{v}$ with approximately $\sqrt{n}$ non-zero entries. Now, we can instantiate the previous idea, but i remains somewhat hidden, i.e., from the receiver's view, it is random among the approximately $n - \sqrt{n}$ zero entries.

To minimize the leakage, we employ a technique known as *biased sampling* as a potential solution used in PIR protocols. In such protocols, the PIR client punctures an alternative index i^* with *a small probability*, rather than the actual index i. This introduces an additional layer of randomness that makes it harder for the PIR server to infer the correct index i. Once the PIR client receives the answer to its query, it can decide whether to use or drop it because the client knows whether a true or an alternative index was punctured. While this technique introduces a minor error in correctness (since i^* is not the actual index), this can be addressed by running the protocol multiple times in parallel and reducing the error rate.

However, in the case of private laconic OT, biased sampling and parallel repetition are insufficient to guarantee index privacy. This is mainly because the roles of the sender and receiver are reversed. During parallel repetition, the receiver can simply focus on the session where a valid message is yielded and infers the information about i from the punctured vector $\mathbf{v}_{-i}$.

To address this limitation, we introduce parallel repetition along with *secret sharing* as a complementary approach to ensure privacy and correctness. In this approach, the sender secret-shares the messages m_0 and m_1 across multiple independent sessions. Each session uses a different punctured vector $\mathbf{v}_{-i}$ by bias sampling and encrypts the shares for that session, making it hard for the receiver to infer information from any single session to determine the sender's index. For correctness, the sender

generates the shares such that in the session where the vector $\mathbf{v}_{-i}$ was punctured at an alternative index, the shares for m_0 and m_1 are identical. In these cases, the receiver still recovers the correct shares for m_0 or m_1 without knowing which one it is. Along with the shares computed from the session where $\mathbf{v}_{-i}$ is punctured at the index i, the receiver can recover the message m_{DB_i}.

Generating Correlations. Up to this point, we have assumed that the sender directly learns the correlation for his chosen index i. Next, we introduce a method to generate the correlations such that the sender must obtain a set of vectors with inner product correlations, ensuring that for each index in the database, at least one vector has a non-zero entry at that index. The challenges are constructing these vectors so that the representation and transmission of these vectors require less space than storing all entries individually and to enable efficient retrieval of the corresponding vector for any given index i.

We leverage the set representation from [CK20] and represent a vector $\mathbf{v}$ by a pseudorandom permutation (PRP) key and a pseudorandom function (PRF) key as follows: First, using the PRP key sk_0, we can generate a set of $\sqrt{n}$ elements, where $S = \{\mathsf{PRP}(\mathsf{sk}_0, 1), \ldots, \mathsf{PRP}(\mathsf{sk}_0, \sqrt{n})\}$, representing the indices with non-zero values. Next, the PRF key sk_1 is used to assign the non-zero values to these indices, with each value computed as $v_x \leftarrow \mathsf{PRF}(\mathsf{sk}_1, x)$ for all $x \in S$. To determine if an index i belongs to the set S, we check if $\mathsf{PRP}^{-1}(\mathsf{sk}_0, i) \leq \sqrt{n}$.

We adopt the "shifting" feature from [CK20] to generate "shifted" vectors. This feature allows the underlying set S, containing the indices of non-zero values, to be shifted by a random shift $\delta \in [n]$. Importantly, only the indices are shifted; the corresponding values are then recomputed using the PRF based on the new, shifted indices. By precomputing $\sqrt{n} \cdot \log n$ random shifts, we ensure that for any index $i \in [n]$, there exists a shift δ_j such that the vector $\mathbf{v}$ shifted by δ_j has a non-zero entry at the index i with non-negligible probability.

Private Laconic OT with Preprocessing. Based on all the previous discussions, we are now in a position to present a full protocol, which consists of the following algorithms:

- A *setup algorithm* that takes as input the security parameter and outputs some public parameters pp.
- An *offline algorithm* that takes the public parameters pp and a database DB as input and outputs the correlation cor.
- An *online sender algorithm* that takes as input the public parameters pp, the correlation cor, an index i, and a pair of messages m_0, m_1, and outputs a ciphertext e.
- An *online receiver algorithm* that takes as input the public parameters pp, the database DB, and the ciphertext e, and outputs a message m.

We illustrate the execution flow of our private laconic OT with a database-dependent preprocessing scheme in the two-server setting in Figure 1. In our construction, the offline server prepares a PRP key sk_0, a PRF key sk_1 and $m = \sqrt{n} \cdot \log n$ random shifts $\delta_1, \ldots, \delta_m \in [n]$. Using these keys and shifts, the offline server generates the initial vector $\mathbf{v}$ and the "shift" vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$ as discussed before. Each "shifted" vector $\mathbf{v}_j$ is then associated with the inner product $\mathsf{ip}_j = \langle \mathbf{v}_j, \mathsf{DB} \rangle$.

The offline server runs this process independently λ times. At the end of the offline phase, it sends λ sets of correlations to the sender. Each correlation set is of the form $\big(\mathsf{sk}_0, \mathsf{sk}_1, \{\delta_j, \langle \mathbf{v}_j, \mathsf{DB} \rangle\}_{j \in [m]}\big)$. These λ sets will be used in the online phase, with each set corresponding to a different session. In our instantiation, the PRP key sk_0 and PRF key sk_1 can be generated once and reused for λ sessions by appending the session index to the keys.

In the online phase, after the sender determines the index i and a pair of messages m_0, m_1, the following procedure is repeated λ times, using the corresponding λ correlations for each session. For simplicity, the session ID and correlation index are omitted from the steps below:

1. The sender chooses the pair (δ, ip) from the corresponding session's correlation, such that $\mathsf{PRP}^{-1}(\mathsf{sk}_0, i - \delta) \leq \sqrt{n}$, which ensures that the chosen shift and inner product correspond to the index i.
2. Next, the sender generates the underlying initial set S using the PRP key sk_0 from this session's correlation and then derives a "shifted" set S' by applying the shift δ. The vector $\mathbf{v}$, used in the inner product computation for ip, is then reconstructed using the set S' and the PRF key sk_1 from the same session's correlation.

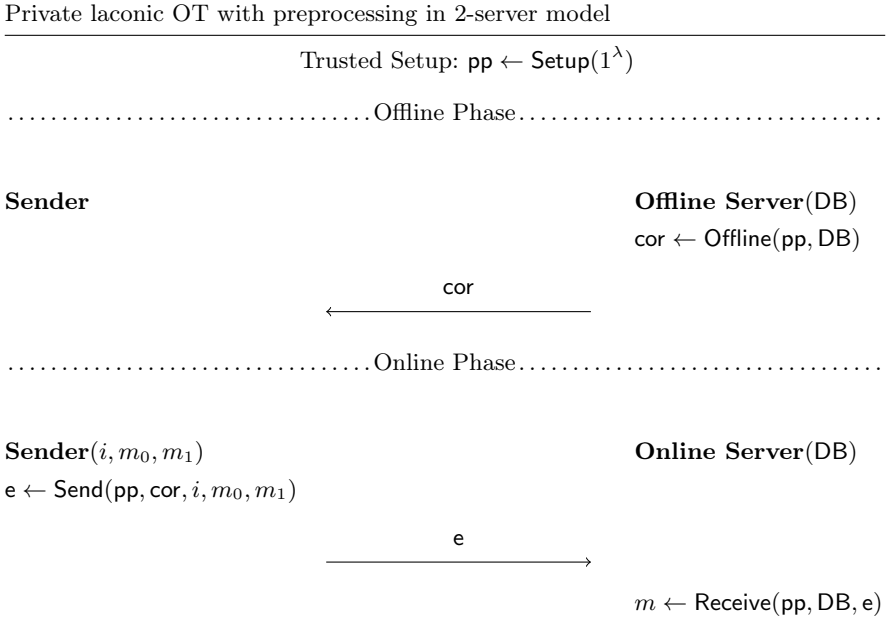

Fig. 1: An illustration of laconic OT with preprocessing in two-server model

3. The sender removes i from S' to create a punctured set of size $\sqrt{n}-1$. But with a small probability, a different index is removed instead of i. The set with i removed is referred to as a correctly punctured set, and the set with a different index removed is called a wrongly punctured set.
4. The sender prepares an additional ordered set that contains the corresponding non-zero values for the indices in the punctured set, generated using the PRF key in the same session's correlation. Each pair consisting of a set of indices and its corresponding ordered set of values is called a set pair.
5. The sender generates two keys $k_0 = \langle \mathbf{v}, \mathsf{DB} \rangle$ and $k_1 = \langle \mathbf{v}, \mathsf{DB} \rangle - v_{i^*}$ where i^* is the actual index removed in Step 3.

After completing these steps, the sender holds λ set pairs and key pairs, with each set pair and key pair corresponding to a session. The sender then generates additive secret shares for m_0 and m_1, using a λ-out-of-λ additive secret shares. Each pair of shares with be associated with the corresponding set pair generated earlier for its respective session. The shares are generated such that, for wrongly punctured sets, the shares for m_0 and m_1 are identical, ensuring that the receiver can infer no useful information. Finally, he encrypts the share pairs with key pairs respectively. The final message e from the sender to the receiver consists of λ set and ciphertext pairs.

Upon receiving e, in the λ sessions, the receiver (online server) recovers the punctured vector $\mathbf{v}'$ for each session. This is done by initializing a zero vector, then using the first set in the set pair to determine which indices should have non-zero values, and then assigning those corresponding values from the second ordered set. Then she computes the key by $k = \langle \mathbf{v}', \mathsf{DB} \rangle$, with which she can decrypt both ciphertexts, and only one of them will yield a valid plaintext that will be the share for that session. After obtaining the shares from all λ sessions, she sums up all the λ shares and gets the message.

In each session, bias sampling makes it harder for the receiver to infer the correct index i. While there is a small probability that the vector (or the underlying set of indices with non-zero values) is wrongly punctured in any individual session, running itλ times drives down the probability to negligible that no correctly punctured vector will occur. Since each session outputs a valid random share, the receiver cannot distinguish the correctly or wrongly punctured set based on the observed shares. This approach addresses the earlier challenge of directly importing biased sampling and parallel repetition from the PIR domain into our setting.

2.3 Extensions and Applications

Updatable Private Laconic OT. We further show our laconic OT correlations can be naturally updated. Namely, when a bit of the database DB changes, the cor for the database can be partially updated rather than regenerated from scratch. Briefly, given a location i that requires an update and a pair of correlations $(\mathbf{v}, y = \langle \mathbf{v}, \mathsf{DB} \rangle)$ such that $v_i \neq 0$. Once the i-th entry in the database is changed, the offline server can update the new correlation as follows

$$y^* = y + (\mathsf{DB}_i^* - \mathsf{DB}_i) \cdot v_i$$

where DB_i^* is the new entry to be written at the i-th position of the database.

Receiver Privacy. In the previous discussion, we do not require receiver privacy as opposed to standard oblivious transfer. Now, we show that our construction can be simply enhanced with receiver privacy against semi-honest senders. We recall that the first laconic OT protocol [CDG+17] achieves receiver privacy by having the two parties run the algorithm Send via a two-round secure 2PC protocol, which can be instantiated via a two-message OT protocol and garbled circuits. For our construction, however, this can be done by secretly sharing the correlations given to the sender.

For this purpose, the offline and online servers hold an extra PRF key k for a PRF f_{k}. During the offline phase, for each correlation indexed by $j \in [m]$ in the correlation sequence, instead of giving $\langle \mathbf{v}, \mathsf{DB} \rangle$ to the sender, the sender only gets a share $a = \langle \mathbf{v}, \mathsf{DB} \rangle - f_{\mathsf{k}}(j)$. Then, in the online phase, once the sender has determined his input, he takes the correlation $(\mathbf{v}, a)$ indexed by l in the correlation sequence and sets the two keys $k_0 = a$ and $k_1 = a - v_i$. He then encrypts his two messages with the two keys and sends the punctured key $\mathbf{v}_{-i}$, two ciphertexts, and the correlation index l to the receiver. The receiver reveals the key by computing $k = \langle \mathbf{v}_{-i}, \mathsf{DB} \rangle - f_{\mathsf{k}}(l)$ and decrypts as before.

Without knowledge of the PRF key k, each share of correlations looks uniformly random to the sender, thus achieving receiver privacy. Furthermore, in our two-server setting, revealing the correlation index l to the receiver does not reveal any information about i or messages m_0, m_1 to the receiver.

Laconic Function Evaluation for RAM Programs Our construction of LFE for RAM programs *with preprocessing* proceeds along the same lines as constructions of RAM-LFE *without preprocessing* [CDG+17, DHMW24]. The birds-eye view of this construction is roughly this: Assume we have laconic OT correlations between a sender and receiver with respect to a receiver database DB. Assume the sender holds a RAM program P with runtime T. Assume for now that P is a read-only RAM program. Such a RAM program can be represented by step-circuits $\mathsf{C}_1, \ldots, \mathsf{C}_T$, where each C_i makes a single query to DB. To encrypt this program P, we will garble *augmented versions* C'_i of the step circuits C_i. The augmented circuit C'_i will receive a preprocessed laconic OT correlation as part of its hardwired input. When the underlying circuit C_i wants to query DB at an index ind, C'_i consumes this correlation to compute a laconic OT for index ind which transfers input labels for the garbling of the next step circuit C'_{i+1}. The final circuit C'_T simply outputs the result of the RAM computation. We can augment this construction to LFE for RAM programs which can perform both read and write queries by simulating an additional writable RAM memory via garbled RAM [GLO15, HKO22]. Furthermore, by hardwiring the laconic OT correlations in their expanded form in the augmented circuits C'_i, we can avoid costly non-black box operations needed to puncture PRF keys in the laconic OT sender algorithm Send.

Laconic Private Set Intersection. Private laconic OT enables the construction of a private membership test. In this test, a receiver holding a dataset X and a sender holding an element y want to jointly compute whether y is in the set X. We now present a private membership test protocol based on our private laconic OT with preprocessing that achieves receiver privacy.

Given a collision-resistant hash function h that maps from the universe to the range $[n]$ (where n is the size of the database), the offline and online servers initialize a database where all coordinates corresponding to elements (mapped by h) are set to 1. The offline server then generates the correlations based on this database and sends them to the sender.

The sender maps his private element y to the corresponding index i using the same hash function h. He then sets m_0 as a random string and m_1 as the actual element prefixed with zeros to ensure

uniform length. Using these inputs, the sender runs the laconic OT Send algorithm with the index i and associated messages m_0 and m_1.

Next, the online server (receiver) performs our laconic OT $\mathsf{Receive}$ algorithm. Since h is assumed to be collision-resistant, the probability of two distinct elements mapping to the same index is negligible. If and only if the receiver's element matches the element mapped to a database entry set to 1, she can decrypt the ciphertexts correctly and reveal m_1, which signals the matched element.

The security against semi-honest senders follows directly from the privacy of the receiver of the laconic OT scheme, and the security against semi-honest receivers follows from the sender's privacy. To extend this private membership test to a laconic private set intersection (PSI) protocol, the sender simply repeats the online procedure for all elements in his set.

3 Cryptographic Preliminaries

3.1 Notations

We use λ to denote the security parameter. We use bold lower-case letters such as $\mathbf{v}$ to represent row vectors, v_i to denote the i-th coordinate of $\mathbf{v}$, and the punctured vector $\mathbf{v}_{-i}$ to represent the vector identical to $\mathbf{v}$ with the exception that $v_i = 0$.

We use $\mathbb{N}$ to denote the set of positive integers. We denote by $[n]$ the set $\{1, \ldots, n\}$ for an integer $n \in \mathbb{N}$, and by $[a, b]$ the set $\{a, \ldots, b\}$ for $a, b \in \mathbb{N}$ and $a < b$. We use $\|$ to denote a concatenation of two binary strings.

We use $x \leftarrow\!\!\$\ S$ to denote sampling x uniformly at random from a finite set S and $x \leftarrow\!\!\$\ \mathcal{D}$ to denote sampling x according to the distribution $\mathcal{D}$. For $p \in [0, 1]$, the notation $b \leftarrow\!\!\$\ \mathsf{Bernoulli}(p)$ refers to choosing the bit b as 1 with probability p and as 0 with probability $1 - p$.

We denote by $\mathsf{negl}(\cdot)$ a negligible function in its parameter, and $\mathsf{poly}(\cdot)$ a fixed polynomial in its parameter. In this work, logarithms are taken to the base 2, and we treat expressions, e.g., $\sqrt{n}$, $\log n$, and m/n as integers, ignoring integrality.

3.2 Private-Key Encryption

In our work, we instantiate private-key (or symmetric-key) encryption scheme with IND-CPA security (Indistinguishability of ciphertext under chosen plaintext attack). A private-key encryption scheme consist of message space $\mathcal{M}$, key space $\mathcal{K}$, ciphertext space $\mathcal{C}$ and three algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. We define the private-key encryption scheme in Definition 1.

Definition 1 (Private-Key Encryption with IND-CPA Security)). *A private-key encryption scheme consist of message space $\mathcal{M}$, key space $\mathcal{K}$, ciphertext space $\mathcal{C}$ and three algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with following syntax and security properties.*

- $k \leftarrow \mathsf{KeyGen}(1^\lambda)$*: Given the security parameter 1^λ, it outputs a key $k \in \mathcal{K}$.*
- $\mathsf{ctxt} \leftarrow \mathsf{Enc}_k(m)$*: Given a message m and key k, it outputs a ciphertext $\mathsf{ctxt} \in \mathcal{C}$.*
- $m \leftarrow \mathsf{Dec}_k(\mathsf{ctxt})$*: Given a key k and ciphertext ctxt, it outputs a message $m \in \mathcal{M}$.*

Correctness: *For every $\lambda \in \mathbb{N}$, all $m \in \mathcal{M}$ and for all $k \leftarrow \mathsf{KeyGen}(1^\lambda)$, it holds that:*

$$m = \mathsf{Dec}_k(\mathsf{Enc}_k(m)).$$

IND-CPA security: *For every $\lambda \in \mathbb{N}$ and for any PPT adversary $\mathcal{A}$ has at most a negligible $\mathsf{negl}(n)$ advantage in the following game played against a challenger:*

1. *The challenger runs $k \leftarrow \mathsf{KeyGen}(1^\lambda)$ and samples a bit $b \in \{0, 1\}$.*
2. *$\mathcal{A}$ sends a pair of message $m_0, m_1 \in \mathcal{M}$ to the challenger and the challenge sends a ciphertext $\mathsf{ctxt} = \mathsf{Enc}_k(m_b)$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs a bit $b^* \in \{0, 1\}$ and $\mathcal{A}$ wins if $b^* = b$.*

3.3 Garbled Circuits

A garbled circuit is an essential cryptographic tool used to evaluate a functionality jointly between two parties.

Definition 2 (Garbled Circuits). *A garbling scheme consists of two algorithms* $(\mathsf{Garble}, \mathsf{Eval})$ *with the following syntax, correctness and security properties.*

- $(\mathsf{C}, \{\mathsf{lb}_{\mathsf{i},b}\}) \leftarrow \mathsf{Garble}(1^\lambda, C)$: *Given the security parameter* λ *and circuit* $C : \{0,1\}^\ell \to \{0,1\}^n$*, it outputs the garbled circuit* C *along with all the input labels of the garbled circuit* $\{\mathsf{lb}_{\mathsf{i},b}\}$*.*
- $y \leftarrow \mathsf{Eval}(\mathsf{C}, \{\mathsf{lb}_\mathsf{i}\})$: *Given the garbled circuit* C *and a label for each input wire denoted by* $\{\mathsf{lb}_i\}$*, it outputs a value* y*.*

We also require the following properties to be satisfied:

Correctness: *For all circuits* C *and input* $x \in \{0,1\}^\ell$*:*

$$\Pr\left[\mathsf{Eval}(\mathsf{C}, \{\mathsf{lb}_{i,x[i]}\}) = C(x) \;\middle|\; (\mathsf{C}, \{\mathsf{lb}_{\mathsf{i},b}\}) \leftarrow \mathsf{Garble}(1^\lambda, C)\right] = 1$$

Security: *For all circuits* C *and input* $x \in \{0,1\}^\ell$*, there exists a simulator such that on input the security parameter* λ *and output value* y *and outputs a simulated garbled circuit* $\hat{\mathsf{C}}$ *and corresponding simulated label values* $\{\hat{\mathsf{lb}}_{\mathsf{i},b}\}$*, and it holds that:*

$$\{(\mathsf{C}, \{\mathsf{lb}_{i,x[i]}\} \mid (\mathsf{C}, \{\mathsf{lb}_{\mathsf{i},b}\}) \leftarrow \mathsf{Garble}(1^\lambda, \mathcal{C})\} \overset{c}{\approx} \{(\hat{\mathsf{C}}, \{\hat{\mathsf{lb}}_{i,x[i]}\} \mid (\hat{\mathsf{C}}, \{\hat{\mathsf{lb}}_{\mathsf{i},b}\}) \leftarrow \mathsf{Sim}(1^\lambda, y)\}$$

4 Database-Dependent Preprocessing

In this section, we describe the offline/online two-server model and the necessity of database-dependent preprocessing for our laconic OT protocol.

4.1 Offline/Online Two-server Model

We model our laconic OT with preprocessing in the offline/online two-server setting. In this setting, there are three parties: a client, an offline server, and an online server. We refer to the online server that receives the OT output as the receiver and the client as the sender. Both the offline and the online server hold a replica of a database DB, but they are not allowed to interact or collude, as established in the model in [BGKW88, CK20].

During the offline phase, the offline server locally generates correlations or preprocessed data based on the database and sends them to the sender. In the online phase, the sender uses these preprocessed correlations to interact with the receiver. The security of the model crucially relies on the offline server and the online server (OT receiver) not communicating or colluding with each other.

4.2 On the Necessity of Database-Dependent Preprocessing

In this part, we will argue that laconic OT with database-independent preprocessing, where the online phase has at most two rounds, implies a standard OT protocol. Hence, it is unlikely that there exists a laconic OT with database-independent preprocessing from symmetric assumptions alone. In particular, such a protocol would need to overcome classical and newer black-box impossibility results [IR90, GHMM18].

We observe that we can strengthen this result by showing that it still holds even if the preprocessing step is allowed to use arbitrarily strong public key primitives. The reason is simple: In our transformed protocol, the preprocessing is entirely performed by the sender, i.e., the receiver runs none of the preprocessing algorithms. In other words, in the transformed protocol, the receiver still only uses symmetric key algorithms, which effectively means it does not matter whether the public key primitives used by the sender were secure or insecure, i.e., the transformed protocol would still have to be secure even if we used an insecure instantiation of these public key primitives.

We will establish these results in the following steps.

1. If the protocol we start with has receiver privacy, we will ignore this feature, i.e., consider it a protocol without receiver privacy.
2. Since the preprocessing phase is independent of the database and as there is no receiver privacy, we can collapse the preprocessing into a single message from the sender to the receiver.
3. We will argue that such a protocol, i.e., laconic OT with a one-message database-independent preprocessing, implies a standard OT protocol with *weak receiver security*.
4. Such a protocol can be amplified into an OT protocol with standard security against semi-honest senders.

Steps 1 and 2 are trivial. We will hence focus on Step 3 and Step 4.

In the following, since this is only a complementary result, we will assume familiarity with standard notions concerning oblivious transfer and forgo providing the most formal definitions.

As mentioned above, we will consider a weak notion of receiver privacy. Specifically, we will consider a setting where the receiver's choice bit is chosen at random, and security only guarantees that the receiver's choice bit is not fully determined by the view of the sender, i.e., the choice bit has some residual Shannon entropy given the view of the sender. A body of works considers the problem of constructing OT protocols with standard security from weakly secure protocols, e.g., [BCW03]. In a nutshell, one can amplify residual Shannon entropy to min-entropy by parallel repetition and then extract a uniformly random choice bit via standard privacy amplification techniques. We will use the following notion of receiver privacy.

Definition 3 (Weak Receiver Privacy). *Let* OT *be an OT protocol with a sender* S *and a receiver* R*, and let* $\phi > 0$*. We say an OT protocol satisfies* ϕ*-weak receiver privacy, if* $H(b|\mathsf{view}(S)) \geq \phi$*. Here*

$$H(X|Y) = -\sum_{x,y} \Pr[X = x, Y = y] \log \left(\frac{\Pr[X = x, Y = y]}{\Pr[X = x]} \right)$$

is the conditional Shannon entropy.

We will finally turn to Step 3, constructing an OT protocol with weak receiver privacy. The idea of this transformation is simple: The receiver chooses a uniformly random database DB, and the sender chooses a random index i. As the first round laconic OT message lOT_1 is short, the entropy of DB cannot be reduced by a lot given lOT_1, i.e., it cannot encode too much information about DB. Even though the first laconic OT message lOT_1 may somehow reveal the bit DB_i for a wrong choice of i, we can show that since i is chosen at random, it holds that the bit DB_i has high (Shannon) entropy *on average*.

We will thus focus on Step 3. Let $\mathsf{lOT} = (\mathsf{Offline}, \mathsf{Rec}_1, \mathsf{Send}, \mathsf{Rec}_2)$ be a 2-message laconic OT protocol with *database-independent* preprocessing $\mathsf{Offline}$. Consider the following OT protocol $\mathsf{OT} = (\mathsf{Send}_1, \mathsf{Rec}_1, \mathsf{Send}_2, \mathsf{Rec}_2)$.

- $\mathsf{Send}_1(1^\lambda, n)$: Compute $(\mathsf{st}_{\mathsf{Send}}, \mathsf{st}_{\mathsf{Rec},1}) = \mathsf{Offline}(1^\lambda, n)$. Output $\mathsf{st}_{\mathsf{Send}}$ and $\mathsf{st}_{\mathsf{Rec},1}$.
- $\mathsf{Rec}_1(\mathsf{st}_{\mathsf{Rec},1})$: Choose $\mathsf{DB} \leftarrow \{0,1\}^n$ uniformly at random. Compute $(\mathsf{lOT}_1, \mathsf{st}_{\mathsf{Rec},2}) = \mathsf{lOT}.\mathsf{Rec}_1(\mathsf{st}_{\mathsf{Rec},1}, \mathsf{DB})$ and output lOT_1 and $\mathsf{st}_{\mathsf{Rec},2}$.
- $\mathsf{Send}_1(\mathsf{st}_{\mathsf{Send}}, m_0, m_1)$: Choose $i \leftarrow [n]$ uniformly at random. Compute and output $\mathsf{lOT}_2 = \mathsf{lOT}.\mathsf{Send}(\mathsf{st}_{\mathsf{Send}}, m_0, m_1)$.
- $\mathsf{Rec}_2(\mathsf{st}_{\mathsf{Rec},2}, \mathsf{lOT}_2)$: Compute $b = \mathsf{DB}_i$ and $m^* = \mathsf{lOT}.\mathsf{Rec}_2(\mathsf{st}_{\mathsf{Rec},2}, \mathsf{lOT}_2)$. Output b, m^*.

We will briefly and informally argue the correctness and sender-privacy of this protocol. Clearly, by the correctness of lOT it holds that $m^* = m_{\mathsf{DB}_i} = m_b$. Moreover, by the sender-privacy of lOT it holds that lOT_2 can be simulated given only $m^* = m_{\mathsf{DB}_i} = m_b$.

We will now establish weak receiver security of our protocol above.

Lemma 1 *The protocol* OT *above satisfies* $1 - \frac{k}{n}$ *weak receiver security.*

Proof. To establish weak receiver privacy, we must show that the receiver's choice bit $b = \mathsf{DB}_i$ has non-trivial Shannon entropy given the sender's view. The view of the sender consists of the

database-independent preprocessing $\mathsf{st}_{\mathsf{Send}}$, as well as the first laconic OT message lOT_1. By the Shannon-entropy chain rule, it holds that

$$\begin{aligned} H(\mathsf{DB}|\mathsf{lOT}_1, \mathsf{st}_{\mathsf{Send}}) &= H(\mathsf{DB}, \mathsf{lOT}_1|\mathsf{st}_{\mathsf{Send}}) - H(\mathsf{lOT}_1|\mathsf{st}_{\mathsf{Send}}) \\ &\geq H(\mathsf{DB}) - k \\ &= n - k \end{aligned}$$

as $\mathsf{st}_{\mathsf{Send}}$ is independent of DB and $|\mathsf{lOT}_1| \leq k$.

By the Shannon-entropy chain rule, it holds that

$$\begin{aligned} H(\mathsf{DB}|\mathsf{lOT}_1, \mathsf{st}_{\mathsf{Send}}) &= H(\mathsf{DB}_1, \ldots, \mathsf{DB}_n|\mathsf{lOT}_1, \mathsf{st}_{\mathsf{Send}}) \\ &= \sum_{j=1}^{n} H(\mathsf{DB}_j|\mathsf{DB}_1, \ldots, \mathsf{DB}_{j-1}, \mathsf{lOT}_1, \mathsf{st}_{\mathsf{Send}}) \\ &\leq \sum_{j=1}^{n} H(\mathsf{DB}_j|\mathsf{lOT}_1, \mathsf{st}_{\mathsf{Send}}) \\ &= n \cdot \sum_{j=1}^{n} \frac{1}{n} H(\mathsf{DB}_j|\mathsf{lOT}_1, \mathsf{st}_{\mathsf{Send}}) \\ &= n \cdot \sum_{j=1}^{n} \Pr[i = j] H(\mathsf{DB}_i|\mathsf{lOT}_1, \mathsf{st}_{\mathsf{Send}}, i = j) \\ &= n \cdot H(\mathsf{DB}_i|\mathsf{lOT}_1, \mathsf{st}_{\mathsf{Send}}, i). \end{aligned}$$

Hence, we obtain that

$$\begin{aligned} H(\mathsf{DB}_i|\mathsf{lOT}_1, \mathsf{st}_{\mathsf{Send}}, i) &\geq \frac{1}{n} \cdot H(\mathsf{DB}|\mathsf{lOT}_1, \mathsf{st}_{\mathsf{Send}}) \\ &\geq \frac{n-k}{n} = 1 - \frac{k}{n}. \end{aligned}$$

□

5 Private Laconic OT with Preprocessing

This section presents our definition and construction of *private laconic OT with preprocessing*. Private laconic OT [DGI+19] inherits all the properties and features of Laconic OT with an additional privacy measure, where the queried index on the database i should not be revealed or leaked to the receiver. Therefore, the message m_{DB_i} received by the receiver does not reveal which message, m_0 or m_1, the receiver has received, which differs from the traditional definition of OT. Our construction follows the model of two servers, where they have the same copy of the database $\mathsf{DB} \in \{0,1\}^n$, but do not interact with each other. One server interacts with the sender in the offline phase, while the other server, acting as the OT receiver, interacts with the sender in the online phase.

5.1 Definition

In this work, to achieve sublinear online time, we define our private laconic OT with preprocessing as follows:

Definition 4 (Private Laconic Oblivious Transfer with Preprocessing). *A private laconic OT with a preprocessing scheme consists of four algorithms* $(\mathsf{Setup}, \mathsf{Offline}, \mathsf{Send}, \mathsf{Receive})$ *with the following syntax, correctness, security, and efficiency properties.*

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*: Given the security parameter* 1^λ*, it outputs a public parameter* pp*.*
- $\mathsf{cor} \leftarrow \mathsf{Offline}(\mathsf{pp}, \mathsf{DB})$*: Given the public parameter* pp *and a database* $\mathsf{DB} \in \{0,1\}^n$*, it outputs the correlation* cor*.*
- $\mathsf{e} \leftarrow \mathsf{Send}(\mathsf{pp}, \mathsf{cor}, i, m_0, m_1)$*: Given the public parameter* pp*, the correlation* cor*, a database location* $i \in [n]$ *and a pair of messages* m_0, m_1 *of length* λ*, it outputs a ciphertext* e*.*

– $m \leftarrow \mathsf{Receive}(\mathsf{pp}, \mathsf{DB}, \mathsf{e})$*: Given the public parameter* pp*, the database* DB *and a ciphertext* e*, it outputs a message* m*.*

Correctness: *For all* $\lambda \in \mathbb{N}$*,* $n = \mathsf{poly}(\lambda)$ *for any polynomial function* $\mathsf{poly}(\cdot)$*, any database* $\mathsf{DB} \in \{0,1\}^n$ *and any pair of messages* $(m_0, m_1) \in \{0,1\}^\lambda \times \{0,1\}^\lambda$*, it holds that*

$$\Pr\left[m = m_{\mathsf{DB}_i} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{cor} \leftarrow \mathsf{Offline}(\mathsf{pp}, \mathsf{DB}) \\ \quad \mathsf{e} \leftarrow \mathsf{Send}(\mathsf{pp}, \mathsf{cor}, i, m_0, m_1) \\ m \leftarrow \mathsf{Receive}(\mathsf{pp}, \mathsf{DB}, \mathsf{e}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Sender Privacy Against Semi-honest Receivers: *For all* $\lambda \in \mathbb{N}$*,* $n = \mathsf{poly}(\lambda)$ *for any polynomial function* $\mathsf{poly}(\cdot)$*, any database* $\mathsf{DB} \in \{0,1\}^n$*, any location* $i \in [n]$ *and any pair of messages* $(m_0, m_1) \in \{0,1\}^\lambda \times \{0,1\}^\lambda$ *there exists a PPT simulator* Sim*, such that:*

$$(\mathsf{pp}, \mathsf{e}) \stackrel{c}{\approx} (\mathsf{pp}, \mathsf{Sim}(\mathsf{pp}, \mathsf{DB}, m_{\mathsf{DB}_i}))$$

where $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $\mathsf{e} \leftarrow \mathsf{Send}(\mathsf{pp}, \mathsf{cor}, i, m_0, m_1)$ *and* $\mathsf{cor} \leftarrow \mathsf{Offline}(\mathsf{pp}, \mathsf{DB})$*.*

Efficiency: *The length of* cor*, and the runtime of the online algorithms* Send *and* $\mathsf{Receive}$ *are sublinear in the size of the database* n*.*

We illustrate the execution of the private laconic OT with the database-dependent preprocessing protocol defined above in Figure 1 (Section 2.2) in the two-server setting. The Setup algorithm is run by a trusted party. During the offline phase, the offline server, holding the database DB, generates the correlations cor by the $\mathsf{Offline}$ algorithm and sends them to the sender. In the online phase, once the sender decides his private index i and a pair of messages m_0, m_1, he computes the ciphertext e with Send algorithm and sends the ciphertext e to the online server. The online server, acting as the OT receiver, decrypts the ciphertext and retrieves m using the $\mathsf{Receive}$ algorithm.

5.2 Construction

We present our construction below, as previously explained in Section 2.2.

Construction 1 (Private Laconic OT with preprocessing) *The protocol is parameterized by a security parameter* $\lambda \in \mathbb{N}$ *and database size* $n \in \mathbb{N}$ *and uses (1) a pseudorandom permutation* $\mathsf{PRP} : \mathcal{K}_\lambda \times [\lambda] \times [n] \mapsto [n]$*, (2) a pseudorandom function* $\mathsf{PRF} : \mathcal{K}_\lambda \times [\lambda] \times [n] \mapsto \mathbb{F}_q^*$ *and (3) a symmetric encryption scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *with key space* $\mathbb{F}_q$*. We define* $m = \sqrt{n} \cdot \log n$*.*

– $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*:*
  1. *Output* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*.*

– $\mathsf{cor} \leftarrow \mathsf{Offline}(\mathsf{pp}, \mathsf{DB})$*:*
  1. *Sample a PRP key* $\mathsf{sk}_0 \leftarrow\!\!\$\ \mathsf{PRP}.\mathcal{K}_\lambda$ *and a PRF key* $\mathsf{sk}_1 \leftarrow\!\!\$\ \mathsf{PRF}.\mathcal{K}_\lambda$*.*
  2. *For each* $\gamma \in [\lambda]$*:*
     (a) *Generate a sequence of random shifts* $\delta_1^\gamma, \ldots, \delta_m^\gamma \leftarrow\!\!\$\ [n]$*.*
     (b) *Generate the initial support sets* $S_0 = \bigcup_{\zeta \in [\sqrt{n}]} \mathsf{PRP}(\mathsf{sk}_0, \gamma, \zeta)$ *and a sequence of shifted sets as* $S_j = \{x + \delta_j^\gamma \bmod n \mid x \in S_0\}$ *for* $j \in [n]$*.*
     (c) *Generate* m *vectors* $\mathbf{v}_1^\gamma, \ldots, \mathbf{v}_m^\gamma \in \mathbb{F}_q^n$ *from the sets* $S_1, \ldots, S_m$ *respectively: For each* S_j*, initialize a zero vector* $\mathbf{v}$ [4] *and set the coordinates* $v_x = \mathsf{PRF}(\mathsf{sk}_1, \gamma, x)$ *for all* $x \in S_j$*.*
  3. *Output* $\mathsf{cor} = (\mathsf{sk}_0, \mathsf{sk}_1, \{(\delta_j^\gamma, \langle \mathbf{v}_j^\gamma, \mathsf{DB} \rangle)\}_{\gamma \in [\lambda], j \in [m]})$*.*

– $\mathsf{e} \leftarrow \mathsf{Send}(\mathsf{pp}, \mathsf{cor}, i, m_0, m_1)$*:*
  1. *Parse* $\mathsf{cor} = (\mathsf{sk}_0, \mathsf{sk}_1, \{(\delta_j^\gamma, \mathsf{ip}_j^\gamma)\}_{\gamma \in [\lambda], j \in [m]})$*.*
  2. *Initialize a zero vector* $\mathbf{l}$ *of size* λ *and generate* λ *sets* $S_1, \ldots, S_\lambda$ *as follows. For each* $\gamma \in [\lambda]$*:*
     (a) *Generate a base set* $S_0 = \bigcup_{\zeta \in [\sqrt{n}]} \mathsf{PRP}(\mathsf{sk}_0, \gamma, \zeta)$*.*
     (b) *Find an* $l_\gamma \in [m]$ *such that* $\mathsf{PRP}^{-1}(\mathsf{sk}_0, \gamma, i - \delta_l^\gamma) \leq \sqrt{n}$*.*
     (c) *Sample a choice bit* $b_\gamma \leftarrow\!\!\$\ \mathsf{Bernoulli}\left(\frac{\sqrt{n}-1}{n}\right)$*. If* $b_\gamma = 0$*, then set* $i_\gamma^* = i$ *otherwise sample* $i_\gamma^* \leftarrow\!\!\$\ S_0 \setminus \{i\}$*.*

[4] For simplicity, we omit the index for the vector in this step.

(d) *Set* $S_\gamma = \{x + \delta_l^\gamma \bmod n \mid x \in S_0\} \setminus \{i^*\}$.

3. *Generate* λ*-out-of-*λ *additive secret shares* $\{s_0^\gamma\}_{\gamma\in[\lambda]}$ *and* $\{s_1^\gamma\}_{\gamma\in[\lambda]}$ *for* m_0 *and* m_1 *respectively subject to the constraint that* $s_0^\gamma = s_1^\gamma$ *for* $b_\gamma = 1$. *Specifically,*
   (a) *For* $\gamma \in [\lambda]$ *such that* $b_\gamma = 1$, *sample* $s_0^\gamma = s_1^\gamma \leftarrow\$ \{0,1\}^\lambda$.
   (b) *Set* $s_0^* = \bigoplus_{\{\gamma\in[\lambda]|b_\gamma=1\}} s_0^\gamma$ *and* $s_1^* = \bigoplus_{\{\gamma\in[\lambda]|b_\gamma=1\}} s_1^\gamma$.
   (c) *For* $\gamma \in [\lambda]$ *such that* $b_\gamma = 0$, *sample* s_0^γ *such that* $\bigoplus_{\{\gamma\in[\lambda]|b_\gamma=0\}} s_0^\gamma = m_0 \oplus s_0^*$ *and* s_1^γ *such that* $\bigoplus_{\{\gamma\in[\lambda]|b_\gamma=0\}} s_1^\gamma = m_1 \oplus s_1^*$.
4. *For* $\gamma \in [\lambda]$, *execute the following:*
   (a) *Set two keys* k_0^γ *and* k_1^γ *as* $k_0^\gamma = \mathsf{ip}_{l_\gamma}^\gamma$ *and* $k_1^\gamma = \mathsf{ip}_{l_\gamma}^\gamma - v_{i^*_\gamma}$.
   (b) *Toss a random coin* $\beta \in \{0,1\}$ *and encrypt the share pair as* $\mathsf{ctxt}_0^\gamma = \mathsf{Enc}_{k_\beta^\gamma}(s_\beta^\gamma||0^\lambda)$ *and* $\mathsf{ctxt}_1^\gamma = \mathsf{Enc}_{k_{1-\beta}^\gamma}(s_{1-\beta}^\gamma||0^\lambda)$.
   (c) *Set an ordered set* $S_\gamma^* = \{\mathsf{PRF}(\mathsf{sk}_1, \gamma, x) \mid x \in S_\gamma\}$.
5. *Output* $\mathsf{e} = \{(S_\gamma, S_\gamma^*, \mathsf{ctxt}_0^\gamma, \mathsf{ctxt}_1^\gamma)\}_{\gamma\in[\lambda]}$.

– $m \leftarrow \mathsf{Receive}(\mathsf{pp}, \mathsf{DB}, \mathsf{e})$*:*

1. *Parse* $\mathsf{e} = \{(S_\gamma, S_\gamma^*, \mathsf{ctxt}_0^\gamma, \mathsf{ctxt}_1^\gamma)\}_{\gamma\in[\lambda]}$.
2. *For* $\gamma \in [\lambda]$, *execute the following:*
   (a) *Generate the vector* $\mathbf{v}^\gamma$ *using the set pair* S_γ *and* S_γ^*, *both of size* $\sqrt{n} - 1$. *The set* S_γ *represents the indices with non-zero values, and the set* S_γ^* *contains the corresponding values, ordered to match the indices in* S_γ. *Specifically, for* $i \in [\sqrt{n} - 1]$, *assign* $v_{S_\gamma[i]}^\gamma = S_\gamma^*[i]$.
   (b) *Compute the key* $k^\gamma = \langle \mathbf{v}^\gamma, \mathsf{DB}\rangle$.
   (c) *Check whether* $\mathsf{Dec}_{k^\gamma}(\mathsf{ctxt}_b)$ *is of form* $s^\gamma\|0^\lambda$ *for* $b \in \{0,1\}$ *and if so, store* s^γ.
3. *Output* $m = \bigoplus_{\gamma\in[\lambda]} s^\gamma$.

Correctness. Correctness follows from the underlying correlations, the encryption scheme, and the secret sharing. In each session, the receiver can compute a key $k_{\mathsf{DB}_{i^*}}$ determined by her choice bit DB_{i^*}, where i^* is the actual index punctured by the sender. This key is then used to decrypt the ciphertext and reveal a share for that session. The secret shares are constructed such that if there is at least one correctly punctured session, summing up all shares will result in the message m_{DB_i}. While there is a small probability that the vector is wrongly punctured in an individual session, doing bias sampling λ times in parallel reduces the probability of having no correctly punctured vector to negligible.

Efficiency. We analyze the size of the correlations as a function of the database size n, implying sublinear communication complexity. The online algorithms Send and Receive run in time sublinear in n. By setting $m = \sqrt{n} \cdot \log n$:

– The Offline algorithm outputs a PRF and PRP key of size $\mathsf{poly}(\lambda)$ each, along with $\lambda\sqrt{n}\log n$ random shifts δ and inner products (correlations), each represented by $\log n$ bits and $\log q$ bits, respectively. This results in the communication of $\mathsf{poly}(\lambda) + \sqrt{n}\lambda\log n(\log n + \log q)$ bits from the offline server to the client in the offline phase, achieving offline communication $\tilde{O}_\lambda(\sqrt{n})$.
– The Send algorithm outputs λ sets of size $\sqrt{n}$ with each element being represented as $\log n$ bits, another λ sets of size $\sqrt{n}$ but with each element represented as $\log q$ bits and 2λ ciphertext of size $\mathsf{poly}(\lambda)$ each. This results in the communication of $\mathsf{poly}(\lambda) + \lambda\sqrt{n}(\log q + \log n)$ bits from the sender to the receiver during the online phase, achieving online communication $\tilde{O}_\lambda(\sqrt{n})$.
– The Send algorithm runs in time $\sqrt{n} \cdot \mathsf{poly}(\lambda)$. The client runs Send where the running time is dominated by the PRP,PRF, and PRP^{-1} queries and encrypts the ciphertexts. PRP, PRP^{-1} and PRF oracles/protocol is queried $\sqrt{n}$ times which takes time $\sqrt{n} \cdot \mathsf{poly}(\lambda)$ and generates λ ciphertext that requires time $\mathsf{poly}(\lambda)$. This leads to the online sender time $O_\lambda(\sqrt{n})$.
– The Receive algorithm runs in time $\lambda\sqrt{n} + \mathsf{poly}(\lambda)$. The online server runs Receive, dominated by the λ inner products and λ decryptions. The λ inner product is between DB and a sparse vector of size $\sqrt{n}$, resulting in a running time of $\lambda\sqrt{n}$ while the λ decryptions require $\mathsf{poly}(\lambda)$ time. This leads to the online server time $O_\lambda(\sqrt{n})$.

Sender Security. The following theorem establishes sender security for our private laconic OT with preprocessing in Construction 1.

Theorem 1 *If* PRP *and* PRF *have pseudorandom property and the symmetric encryption scheme* $\mathsf{Sym} = (\mathsf{Enc}, \mathsf{Dec})$ *is IND-CPA secure, then the private laconic OT with preprocessing in Construction 1 guarantees sender privacy against a semi-honest receiver.*

Proof. We show that it is possible to simulate the transcripts from the sender to the receiver which are indistinguishable from the view in a real world. On input the public parameter pp, a database DB and the learned message m, we simulate the view of a corrupted receiver in the following way:

1. Choose a random message $\hat{m} \leftarrow\!\$\ \{0,1\}^\lambda$.
2. For each $\gamma \in [\lambda]$:
   (a) Sample a set $S_\gamma \subset [n]$ of size $\sqrt{n}-1$ representing the indices with non-zero values.
   (b) Sample an ordered set $S^*_\gamma \leftarrow\!\$\ (\mathbb{F}^*_q)^{\sqrt{n}-1}$ representing the non-zero values.
3. Generate λ-out-of-λ additive secret shares $\{s^\gamma\}_{\gamma\in[\lambda]}$ and $\{\hat{s}^\gamma\}_{\gamma\in[\lambda]}$ randomly for m and $\hat{m}$ respectively subject to the constraint that:
   (a) $\bigoplus_{\gamma\in[\lambda-1]} s^\gamma = m \oplus s^\lambda$.
   (b) $\bigoplus_{\gamma\in[\lambda-1]} s^\gamma = \hat{m} \oplus \hat{s}^\lambda$.
4. For each $\gamma \in [\lambda]$:
   (a) Generate the vector $\mathbf{v}^\gamma$ using the set pair S_γ and S^*_γ by assigning $v^\gamma_{S_\gamma[i]} = S^*_\gamma[i]$ for $i \in [\sqrt{n}-1]$ and 0 elsewhere.
   (b) Toss a random coin $\beta^\gamma \in \{0,1\}$ and set $k^\gamma_\beta = \langle \mathbf{v}^\gamma, \mathsf{DB}\rangle$ and $k^\gamma_{1-\beta} \leftarrow\!\$\ \mathbb{F}_q$.
   (c) Compute $\mathsf{ctxt}^\gamma_{\beta^\gamma} = \mathsf{Enc}_{k^\gamma_\beta}(s^\gamma||0^\lambda)$ and $\mathsf{ctxt}^\gamma_{1-\beta^\gamma} = \mathsf{Enc}_{k^\gamma_{1-\beta}}(\hat{s}^\gamma||0^\lambda)$.
5. Output $\mathsf{e} = \{(S_\gamma, S^*_\gamma, \mathsf{ctxt}^\gamma_0, \mathsf{ctxt}^\gamma_1)\}_{\gamma\in[\lambda]}$.

We now show that the simulated view is indistinguishable from the view in the real execution using the following sequence of hybrids.

- Hyb_0: This hybrid proceeds as the real world.
- Hyb_1: Same as Hyb_0 except that the two messages in the real world will be replaced with the input message m and another random message $\hat{m} \leftarrow\!\$\ \{0,1\}^\lambda$. As the shares are generated randomly and by tossing a random coin to switch the position of ciphertexts, this hybrid will be indistinguishable from Hyb_0 due to the semantic security of the encryption scheme.
- Hyb_2: Same as Hyb_1 except that the vector used to compute the keys for the encryption scheme is not reconstructed by the PRP and PRF, but by randomly sampling a vector of length n and hamming weight $\sqrt{n}-1$. By Lemma 2, this is indistinguishable from the previous Hyb_1: It is indistinguishable whether a set, which represents the vector indices with non-zero values, is correctly or wrongly punctured in each session. And since the non-zero values for the indices are generated via a PRF in the real-world case, it is indistinguishable between the PRF generated value and a uniformly random value. This concludes that the freshly generated vector is indistinguishable from the one generated from PRP and PRF.
- Hyb_3: Same as Hyb_2 and note that this is identical to the view of the receiver in the simulation. This conlcues the proof.

Lemma 2 *Given a pseudorandom permutation* $\mathsf{PRP} : \mathcal{K}_\lambda \times [\lambda] \times [n] \mapsto [n]$*, for* $\lambda \in \mathbb{N}$*,* $n = \mathsf{poly}(\lambda)$ *and every* $i \in [n]$*,* $\gamma \in [\lambda]$*, define the following distribution*

$$\mathcal{D}_{\lambda,n,i,\gamma} = \left\{ \begin{array}{l} \mathsf{sk} \leftarrow\!\$\ \mathsf{PRP}.\mathcal{K}_\lambda \text{ s.t. } \mathsf{PRP}^{-1}(\mathsf{sk},\gamma,i) \le \sqrt{n} \\ S = \{\mathsf{PRP}(\mathsf{sk},\gamma,1),\ldots,\mathsf{PRP}(\mathsf{sk},\gamma,\sqrt{n})\} \\ b \leftarrow\!\$\ \mathsf{Bernoulli}\left(\frac{\sqrt{n}-1}{n}\right) \\ \text{if } b = 0 : i^* = i \\ \text{if } b = 1 : i^* \leftarrow\!\$\ S \setminus \{i\} \\ \text{output } S \setminus \{i^*\} \end{array} \right\}.$$

Then, for every $i, j \in [n]$*, it holds that*

$$\mathcal{D}_{\lambda,n,i,\gamma} \overset{c}{\approx} \mathcal{D}_{\lambda,n,j,\gamma}.$$

This lemma is the key result from the PIR work (Lemma 36, [CK20]). It shows that it is possible to sample a PRP key sk for a set S and a punctured set in such a way that a chosen element $i \in [n]$ is in the set S and the punctured set completely hides the chosen point i. We represent the lemma with the syntax of pseudorandom permutation. □

6 Extensions

In this section, we show how to extend our construction to support updatability and receiver privacy, broadening its applicability to a wider range of scenarios.

6.1 Updatability

In many applications of laconic OT, an updatable version is crucial. We present an updatable variant of our private laconic OT with preprocessing, where the correlations cor committed to a database DB can be partially updated when a bit of the database changes, rather than being regenerated entirely.

Definition 5 (Updatable Private Laconic OT with Preprocessing). *An updatable private laconic OT with preprocessing scheme consists of the algorithms* $(\mathsf{Setup}, \mathsf{Offline}, \mathsf{Send}, \mathsf{Receive})$ *as defined in Definition 4 and additionally one algorithm* Upd *with the following syntax.*

- $\langle \mathsf{cor}^*, \mathsf{DB}^* \rangle \leftarrow \mathsf{Upd}(\mathsf{pp}, \mathsf{DB}, \mathsf{cor}, i, b)$*: Given the public parameter* pp*, the database* $\mathsf{DB} \in \{0,1\}^n$*, the correlation* cor*, a location* $l \in [n]$ *and a bit* $b \in \{0,1\}$ *to be written,it outputs an updated correlation* cor^* *to the sender and an updated database* DB^* *to the receiver.*

We require the following properties with regards to update on top of those properties of a private laconic OT with preprocessing scheme.

Correctness w.r.t. Update: *For all* $\lambda \in \mathbb{N}$*,* $n = \mathsf{poly}(\lambda)$ *for any polynomial function* $\mathsf{poly}(\cdot)$*, any database* $\mathsf{DB} \in \{0,1\}^n$*, any location* $i \in [n]$ *and any pair of messages* $(m_0, m_1) \in \{0,1\}^\lambda \times \{0,1\}^\lambda$*, given the public parameter* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*, let* DB^* *be identical to* DB *except that* $\mathsf{DB}^*_i = b$*, the following are from the identical distribution:*

$$\{\mathsf{cor}^* \mid \langle \mathsf{cor}^*, \mathsf{DB}^* \rangle \leftarrow \mathsf{Upd}(\mathsf{pp}, \mathsf{cor}, i, b) \wedge \mathsf{cor} \leftarrow \mathsf{Offline}(\mathsf{pp}, \mathsf{DB})\}$$
$$\{\mathsf{cor} \mid \mathsf{cor} \leftarrow \mathsf{Offline}(\mathsf{pp}, \mathsf{DB}^*)\}$$

Our proposed laconic OT correlation shows that the correlation can be efficiently updated.

Construction 2 (Updating Private OT Correlations) *Given the public parameter* pp*, the database* DB *to be updated, a location* i *to be written, a correlation* cor*, a bit* $b \in \{0,1\}$*, the* Upd *algorithm performs the following steps:*

1. *Parse* $\mathsf{cor} = (\mathsf{sk}_0, \mathsf{sk}_1, \{(\delta_j^\gamma, \mathsf{ip}_j^\gamma)\}_{\gamma \in [n], j \in [m]})$.
2. *For each* $\gamma \in [\lambda]$*:*
   (a) *Find an* $l \in [m]$ *such that* $\mathsf{PRP}^{-1}(\mathsf{sk}_0, \gamma, i - \delta_l^\gamma) \leq \sqrt{n}$.
   (b) *Replace* ip_l^γ *with the following one:*

$$\mathsf{ip}_l^{\gamma *} = \mathsf{ip}_l^\gamma + (b - \mathsf{DB}_i) \cdot \mathsf{PRF}(\mathsf{sk}_1, \gamma, i).$$

   Then get the updated correlations cor^*.
3. *Update the database into* DB^* *such that* $\mathsf{DB}^*_i = b$ *and identical to* DB *at remaining indices.*
4. *Output* $\langle \mathsf{cor}^*, \mathsf{DB}^* \rangle$.

Intuitively, the correctness follows from the randomness of the vectors we sample. We consider the following two cases when we update i-th entry of the database DB to $b \in \{0,1\}$ (session id γ omitted):

- In the real world of Upd, we have the correlation as

$$\langle \mathbf{v}, \mathsf{DB} \rangle + (b - \mathsf{DB}_i) \cdot v_i = \sum_{j \in \mathsf{support}(\mathbf{v})} \mathsf{DB}_j \cdot v_j + (b - \mathsf{DB}_i) \cdot v_i = \langle \mathbf{v}_{-i}, \mathsf{DB} \rangle + b \cdot v_i$$

  where $\mathsf{support}(\mathbf{v})$ represents the set $S = \{\mathsf{PRP}(\mathsf{sk}_0, 1), \ldots, \mathsf{PRP}(\mathsf{sk}_0, \sqrt{n})\}$ and $v_i \leftarrow \mathsf{PRF}(\mathsf{sk}_1, i)$.
- In the ideal world where the correlation is freshly generated based on the updated database DB^* where $\mathsf{DB}_l = b$, we have the correlation

$$\langle \mathbf{v}, \mathsf{DB}^* \rangle = \langle \mathbf{v}_{-i}, \mathsf{DB} \rangle + b \cdot v_i$$

  where $v_i \leftarrow \mathsf{PRF}(\mathsf{sk}_1, i)$.

It is straightforward that the two distributions are indistinguishable and the update algorithm preserves correctness.

6.2 Receiver Privacy

In the definition of private laconic OT with preprocessing provided in Definition 4, we do not require receiver privacy, meaning there is no privacy guarantee against a corrupted sender, as defined in [CDG+17]. Typically, adding receiver privacy to laconic OT can be achieved straightforwardly using garbled circuits and two-message OT. However, we demonstrate that our protocol in the two-server setting can be enhanced with this property without introducing additional costly cryptographic primitives.

Definition 6 (Receiver Privacy). *A private laconic OT with preprocessing scheme defined in Definition 4 provides receiver privacy against semi-honest senders if for all $\lambda \in \mathbb{N}$, $n = \mathsf{poly}(\lambda)$ for any polynomial function $\mathsf{poly}(\cdot)$ and any database $\mathsf{DB} \in \{0,1\}^n$, that there exists a PPT simulator Sim, such that*

$$(\mathsf{pp}, \mathsf{Offline}(\mathsf{pp}, \mathsf{DB})) \stackrel{c}{\approx} (\mathsf{pp}, \mathsf{Sim}(\mathsf{pp}))$$

where $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$.

Construction 3 (Receiver-private plOT with Preprocessing) *Given a pseudorandom function $\mathsf{PRF}^* : \mathcal{K}_\lambda \times [\lambda] \times [m] \mapsto \mathbb{F}_q^n$ with the PRF key $\mathsf{k} \in \mathcal{K}_\lambda$ to the two servers only, the private laconic OT with preprocessing protocol with algorithms $(\mathsf{Setup}, \mathsf{Offline}, \mathsf{Send}, \mathsf{Receive})$ shown in Construction 1 can be extended to that with algorithms $(\mathsf{Setup}^*, \mathsf{Offline}^*, \mathsf{Send}^*, \mathsf{Receive}^*)$ supporting receiver privacy as follows:*

- $\mathsf{pp} \leftarrow \mathsf{Setup}^*(1^\lambda)$*:*
  1. *Output* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
- $\mathsf{cor} \leftarrow \mathsf{Offline}^*(\mathsf{pp}, \mathsf{DB})$*:*
  1. *Compute* $(\mathsf{sk}_0, \mathsf{sk}_1, \{(\delta_j^\gamma, \mathsf{ip}_j^\gamma)\}_{\gamma\in[n], j\in[m]}) \leftarrow \mathsf{Offline}(\mathsf{pp}, \mathsf{DB})$.
  2. *For all* $\gamma \in [\lambda]$*, set* $a_j^\gamma = \mathsf{ip}_j^\gamma - \mathsf{PRF}^*(\mathsf{k}, \gamma, j)$ *for all* $j \in [m]$.
  3. *Output* $\mathsf{cor} = ((\mathsf{sk}_0, \mathsf{sk}_1, \{(\delta_j^\gamma, a_j^\gamma)\}_{\gamma\in[n], j\in[m]}))$.
- $\mathsf{e} \leftarrow \mathsf{Send}^*(\mathsf{pp}, \mathsf{cor}, i, m_0, m_1)$*:*
  1. *Parse* $\mathsf{cor} = ((\mathsf{sk}_0, \mathsf{sk}_1, \{(\delta_j^\gamma, a_j^\gamma)\}_{\gamma\in[n], j\in[m]}))$.
  2. *Proceed with Step 2 to 4 of the* Send *algorithm in Construction 1, except that each* ip *is replaced as a in this case (index j and γ omitted here).*
  3. *Output* $\mathsf{e} = \{(S_\gamma, S_\gamma^*, l_\gamma, \mathsf{ctxt}_0^\gamma, \mathsf{ctxt}_1^\gamma)\}_{\gamma\in[\lambda]}$*, where l_γ is generated in Step 2b of the* Send *algorithm, indicating the index of the used correlation for the γ-th session.*
- $m \leftarrow \mathsf{Receive}^*(\mathsf{pp}, \mathsf{DB}, \mathsf{e})$*:*
  1. *Parse* $\mathsf{e} = \{(S_\gamma, S_\gamma^*, l_\gamma, \mathsf{ctxt}_0^\gamma, \mathsf{ctxt}_1^\gamma)\}_{\gamma\in[\lambda]}$.
  2. *Proceed with Step 2 in the* $\mathsf{Receive}$ *algorithm in Construction 1, except that the key is now computed as* $k^\gamma = \langle \mathbf{v}^\gamma, \mathsf{DB}\rangle - \mathsf{PRF}^*(\mathsf{k}, \gamma, l_\gamma)$.
  3. *Output* $m = \bigoplus_{\gamma\in[\lambda]} s^\gamma$.

Correctness. The correctness of the above modified scheme follows from that of Construction 1. Intuitively, secret sharing the correlations, i.e., inner product computations, does not change the correlation argument at all.

Efficiency. Compared with Construction 1, the efficiency is preserved as well except that the two servers hold extra λ PRF key each of size λ bits, and the size of e is increased by λ correlation indices l_γ each of size λ bits.

Security. The sender privacy is remained the same as in Construction 1 given the correlation index l looks uniformly random to the online server (OT receiver). The following theorem establishes receiver privacy.

Theorem 2 *If the extra PRF^* have pseudorandom property, then the private laconic OT with preprocessing in Construction 3 guarantees receiver privacy against a semi-honest sender.*

Proof. We show that we can simulate the transcript for a semi-honest sender. We define a PPT Sim as follows:

1. On input the public parameters pp, it generates a PRP key $\mathsf{sk}_0 \leftarrow\$ \mathsf{PRP}.\mathcal{K}_\lambda$ and a PRF key $\mathsf{sk}_1 \leftarrow\$ \mathsf{PRF}.\mathcal{K}_\lambda$, and generate a sequence of random shifts $\delta_1^\gamma, \ldots, \delta_m^\gamma \leftarrow\$ [n]$ for $\gamma \in [\lambda]$.

2. For each δ_j^γ, pair it with a fresh sampled $r_j^\gamma \leftarrow\!\!\$\, \mathbb{F}_q$.
3. Output $\mathsf{cor} = ((\mathsf{sk}_0, \mathsf{sk}_1, \{(\delta_j^\gamma, r_j^\gamma)\}_{j\in[m]}, \gamma \in [\lambda]))$.

We now show that the simulated view is indistinguishable from the view in the real execution using the following sequence of hybrids.

- Hyb_0: Identical to the view of the sender in the real protocol.
- Hyb_1: Instead of relying on DB and a PRF^* key k to generate a share for each correlation, we change each share as a randomness $r_j \leftarrow\!\!\$\, \mathbb{F}_q$ for $j \in [m]$. Without the knowledge of PRF^* key k, the distribution of r_j^γ is indistinguishable from that of a_j^γ in the real world. Therefore, Hyb_0 and Hyb_1 is indistinguishable.
- Hyb_2: Identical to the view in the simulation. In Hyb_1, each component from the real protocol has been replaced with that from the simulation. Therefore, Hyb_1 and Hyb_2 are identical. This concludes this proof.

□

7 Laconic Function Evaluation for RAM Programs

In this section, we discuss the main application of our private laconic OT with preprocessing: laconic function evaluation for RAM programs (RAM-LFE) in our two-server model.

7.1 RAM Programs

We will briefly describe the RAM model considered in this paper. A RAM program P with runtime T as a sequence of step-circuits $\mathsf{C}_1, \ldots, \mathsf{C}_T$. The circuits get access to a large input database and a work database, where the latter is initialized with zeros. Each circuit can make a single read query to the input database D, as well as a single read and a single write query to the work database W. More formally, each step-circuit C_i receives as input a state st_i, an input d which is the result of a read-query to D by C_{i-1}, and an input w which is the result of a write-query to W by C_{i-1}. C_i returns a state st_{i+1}, an index ind_D for a read-query on D and an index ind_W for a read-query on W, and a pair $(\bar{\mathsf{ind}}, y)$ that instructs to write symbol y to location $\bar{\mathsf{ind}}$ in database W.

We will make a few additional assumptions about the RAM program P, in particular, that read-queries may fail. In this case, the read-query returns a special symbol $\perp$. We will briefly argue that if the probability of a read-query failing is sufficiently small, then a small increase in runtime can compensate for read errors without changing the overall program structure.

7.2 Construction

We will now present a construction of RAM-LFE for read-only RAM programs. We will then discuss how this can be extended to RAM-LFE for RAM programs that make both read and write requests via efficient garbled RAM schemes.

Definition 7 (RAM-LFE). *A laconic function evaluation for RAM programs (RAM-LFE) consists of the four algorithms* $(\mathsf{Setup}, \mathsf{Offline}, \mathsf{Send}, \mathsf{Receive})$ *with the following syntax, correctness, security and efficiency properties.*

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*: Given the security parameter* 1^λ*, it outputs a public parameter* pp*.*
- $\mathsf{cor} \leftarrow \mathsf{Offline}(\mathsf{pp}, \mathsf{DB}, T)$*: Given the public parameter* pp*, a database* $\mathsf{DB} \in \{0,1\}^n$ *and a runtime* T*, it outputs the correlation* cor*.*
- $\mathsf{e} \leftarrow \mathsf{Send}(\mathsf{pp}, \mathsf{cor}, P)$*: Given the public parameter* pp*, the correlation* cor*, a program* P*, it outputs a ciphertext* e*.*
- $m \leftarrow \mathsf{Receive}(\mathsf{pp}, \mathsf{DB}, \mathsf{e})$*: Given the public parameter* pp*, the database* DB *and a ciphertext* e*, it outputs a value* m*.*

Correctness: *For all* $\lambda \in \mathbb{N}$*,* $n = \mathsf{poly}(\lambda)$ *for any polynomial function* $\mathsf{poly}(\cdot)$*, any database* $\mathsf{DB} \in \{0,1\}^n$ *and any program* P *with runtime* $T = \mathsf{poly}(\lambda)$*, it holds that*

$$\Pr\left[m = P^{\mathsf{DB}} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{cor} \leftarrow \mathsf{Offline}(\mathsf{pp}, \mathsf{DB}, T) \\ \mathsf{e} \leftarrow \mathsf{Send}(\mathsf{pp}, \mathsf{cor}, P) \\ m \leftarrow \mathsf{Receive}(\mathsf{pp}, \mathsf{DB}, \mathsf{e}) \end{array} \right] \geq \mathsf{negl}(\lambda).$$

Sender Privacy Against Semi-honest Receivers: *For all $\lambda \in \mathbb{N}$, $n = \mathsf{poly}(\lambda)$ for any polynomial function $\mathsf{poly}(\cdot)$, any database $\mathsf{DB} \in \{0,1\}^n$, any program P with runtime $T = \mathsf{poly}(\lambda)$, it holds that there exists a PPT simulator Sim, such that*

$$(\mathsf{pp}, \mathsf{Send}(\mathsf{pp}, \mathsf{cor}, P)) \stackrel{c}{\approx} (\mathsf{pp}, \mathsf{Sim}(\mathsf{pp}, \mathsf{DB}, m))$$

where $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{cor} \leftarrow \mathsf{Offline}(\mathsf{pp}, \mathsf{DB}, T)$.

Efficiency: *The length of cor, and the runtime of the algorithm $\mathsf{Offline}$, Send and $\mathsf{Receive}$ are sublinear in the size of the database n.*

The idea for our read-only RAM-LFE follows naturally from combining private laconic OT correlations with garbled circuits: In the offline phase, we generate private laconic OT correlations with respect to the receiver's database DB for each step-circuit C_i, and in the online phase, we hardwire these correlations into an extended version C'_i of C_i to allow access to the database DB.

Hence, let C_i be a step circuit. Assume for convenience that C_i takes as input a state st_i as well as a bit w_i (which is the result of the previous read-query), and outputs a state st_{i+1} and an index ind for a read-query. The circuit C'_i has hardwired inputs $\mathsf{lab}_{\mathsf{st}_{i+1}}$, $\mathsf{lab}_{w_{i+1}}$ and a private laconic OT correlation cor_i. Here, $\mathsf{lab}_{\mathsf{st}_{i+1}}$ are the input labels corresponding to st_{i+1} for a garbling of the next step-circuit C'_{i+1}, whereas $\mathsf{lab}_{w_{i+1}}$ are likewise input labels corresponding to the bit w_{i+1}.

The augmented step circuit c'_i is given as follows.

- Circuit $\mathsf{C}'_i[\mathsf{pp}, \mathsf{lab}_{\mathsf{st}_{i+1}}, \mathsf{lab}_{w_{i+1}}, \mathsf{cor}_i](\mathsf{st}_i, w_i)$:
  - Compute $(\mathsf{st}_{i+1}, \mathsf{ind}_{i+1}) \leftarrow \mathsf{C}(\mathsf{st}_i, w_i)$
  - Set $\mathsf{lb}_{\mathsf{st}_{i+1}} \leftarrow \mathsf{LabelEncoding}(\mathsf{lab}_{\mathsf{st}_{i+1}}, \mathsf{st}_{i+1})$
  - Compute $e_{i+1} \leftarrow \mathsf{Send}(\mathsf{pp}, \mathsf{cor}_i, \mathsf{ind}_{i+1}, \mathsf{lab}_{w_{i+1},0}, \mathsf{lab}_{w_{i+1},1})$
  - Output $(\mathsf{lb}_{\mathsf{st}_{i+1}}, e_{i+1})$

Our RAM-LFE scheme $\mathsf{RLFE} = (\mathsf{RLFE.Setup}, \mathsf{RLFE.Offline}, \mathsf{RLFE.Send}, \mathsf{RLFE.Receive})$ in the preprocessing model is now given as follows.

Construction 4 (RAM-LFE) *Let $(\mathsf{Setup}, \mathsf{Offline}, \mathsf{Send}, \mathsf{Receive})$ be a private laconic OT with preprocessing scheme. Further $(\mathsf{GC.Garble}, \mathsf{GC.Eval})$ be a projective garbling scheme. The scheme $\mathsf{RLFE} = (\mathsf{RLFE.Setup}, \mathsf{RLFE.Offline}, \mathsf{RLFE.Send}, \mathsf{RLFE.Receive})$ is given as follows.*

- $\mathsf{RLFE.Setup}(1^\lambda)$*: Compute and output* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$.
- $\mathsf{RLFE.Offline}(\mathsf{pp}, \mathsf{DB}, T)$*:*
  1. *For $i \in [T]$ compute* $\mathsf{cor}_i \leftarrow \mathsf{Offline}(\mathsf{pp}, \mathsf{DB})$*;*
  2. *Output* $\mathsf{cor}^* \leftarrow (\mathsf{cor}_i)_{i \in [T]}$.
- $\mathsf{RLFE.Send}(\mathsf{pp}, \mathsf{cor}^*, P)$*:*
  1. *Parse* $\mathsf{cor}^* = (\mathsf{cor}_i)_{i \in [T]}$ *and* $P = (\mathsf{C}_1, \ldots, \mathsf{C}_T)$*;*
  2. *Compute* $(\tilde{\mathsf{C}}_T, \mathsf{lab}_{\mathsf{st}_T}, \mathsf{lab}_{w_T}) \leftarrow \mathsf{GC.Garble}(\mathsf{C}_T)$*;*
  3. *For* $i = T-1, \ldots, 1$ *compute* $(\tilde{\mathsf{C}}_i, \mathsf{lab}_{\mathsf{st}_i}, \mathsf{lab}_{w_i}) \leftarrow \mathsf{GC.Garble}(\mathsf{C}_i[\mathsf{lab}_{\mathsf{st}_{i+1}}, \mathsf{lab}_{w_{i+1}}, \mathsf{cor}_i])$*;*
  4. *Set* $\mathsf{lb}_{\mathsf{st}_1} = \mathsf{LabelEncoding}(\mathsf{lab}_{\mathsf{st}_1}, 0^*)$, $\mathsf{lb}_{w_1} = \mathsf{LabelEncoding}(\mathsf{lab}_{w_1}, 0)$*;*
  5. *Output* $e^* = (\mathsf{lb}_{\mathsf{st}_1}, \mathsf{lb}_{w_1}, \tilde{\mathsf{C}}_1, \ldots, \tilde{\mathsf{C}}_T)$.
- $\mathsf{RLFE.Receive}(\mathsf{pp}, \mathsf{DB}, e^*)$*:*
  1. *Parse* $e^* = (\mathsf{lb}_{\mathsf{st}_1}, \mathsf{lb}_{w_1}, \tilde{\mathsf{C}}_1, \ldots, \tilde{\mathsf{C}}_T)$*;*
  2. *For* $i = 1, \ldots, T-1$*:*
     (a) *Compute* $(\mathsf{lb}_{\mathsf{st}_{i+1}}, e_{i+1}) \leftarrow \mathsf{GC.Eval}(\tilde{\mathsf{C}}_i, \mathsf{lb}_{\mathsf{st}_i}, \mathsf{lb}_{w_i})$
     (b) *Compute* $\mathsf{lb}_{w_{i+1}} \leftarrow \mathsf{Receive}(\mathsf{pp}, \mathsf{DB}, e_{i+1})$
  3. *Compute and output* $y \leftarrow \mathsf{GC.Eval}(\tilde{\mathsf{C}}_T, \mathsf{lb}_{\mathsf{st}_T}, \mathsf{lb}_{w_T})$.

Correctness. The correctness of the above scheme RLFE routinely follows from the correctness of the underlying garbling scheme $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Eval})$ and the correctness of the private laconic OT with preprocessing scheme $(\mathsf{Setup}, \mathsf{Offline}, \mathsf{Send}, \mathsf{Receive})$.

Efficiency. The ciphertext size of RLFE is dominated by the hardwired correlations cor_i. In Construction 1, given that the size of cor_i is $\tilde{O}_\lambda(\sqrt{n})$, the total ciphertext size is $\tilde{O}_\lambda(T \cdot \sqrt{n})$. Thus, due to the efficiency properties of the underlying laconic OT with preprocessing scheme, the overhead for sender and receiver will also be $O_\lambda(T \cdot \sqrt{n})$.

Sender Security. The following theorem establishes sender security for RLFE.

Theorem 3 *Assume that* $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Eval})$ *is a simulation secure garbling scheme and that* $\mathsf{laconicOT} = (\mathsf{Setup}, \mathsf{Offline}, \mathsf{Send}, \mathsf{Receive})$ *has sender privacy against semi-honest receivers. Then the scheme* $\mathsf{RLFE} = (\mathsf{RLFE.Setup}, \mathsf{RLFE.Offline}, \mathsf{RLFE.Send}, \mathsf{RLFE.Receive})$ *also has sender privacy against semi-honest receivers.*

Proof. The proof proceeds via a hybrid argument. We will define $2T + 1$ hybrids Hyb_0, $\mathsf{Hyb}_{i,0}$ and $\mathsf{Hyb}_{i,1}$ for $i = 1, \ldots, T$.

- Hyb_0: This is the real experiment.
- $\mathsf{Hyb}_{i,0}$ is identical to $\mathsf{Hyb}_{i-1,1}$ (or Hyb_0 if $i = 1$), except that we compute $\tilde{\mathsf{C}}_i$, $\mathsf{lb}_{\mathsf{st}_i}$ and lb_{w_i} via $(\tilde{\mathsf{C}}_i, \mathsf{lb}_{\mathsf{st}_i}) \leftarrow \mathsf{GC.Sim}(\mathsf{C}_i[\mathsf{pp}, \mathsf{lab}_{\mathsf{st}_{i+1}}, \mathsf{lab}_{w_{i+1}}, \mathsf{cor}_i](\mathsf{st}_i, w_i))$
  Indistinguishability follows by the simulation security of GC.
- $\mathsf{Hyb}_{i,1}$ is identical to $\mathsf{Hyb}_{i,0}$, except that we compute e_{i+1} (which is computed by C_i) via $e_{i+1} \leftarrow \mathsf{Sim}(\mathsf{lb}_{w_{i+1}, \mathsf{DB}_{\mathsf{ind}_{i+1}}})$. Indistinguishability follows by the sender privacy of laconic OT.

Note that in the last hybrid, the output of the receiver depends only on $\mathsf{C}_T(\mathsf{st}_T, w_T)$ (i.e., the correct output of P), but is otherwise independent of P, since all labels used during the simulation are chosen uniformly at random. This concludes the security proof. □

Beyond read-only RAM Programs We will briefly discuss how the RLFE scheme can be upgraded to support RAM programs that support both read and write queries. Instead of relying on updatable laconic OT like previous work [CDG+17,DHMW24], we can combine the scheme RLFE with a garbled RAM scheme [LO13,GHL+14,GLO15,HKO22]. Specifically, such schemes also proceed by successively garbling the step circuits of the RAM, but also provide a garbled data-structure for the work database, which can be both read from and written to. For a garbled work database of size L, these constructions incur only an additive overhead of size $\tilde{O}(L)$ (in addition to the size of the garbled step circuits). Assuming that the work memory is initialized with zeros, we observe that a RAM program running in time T can read from and write to at most T locations of the work memory. Consequently, a work memory of size $O(T)$ is sufficient in this setting. We can obtain a full-fledged RAM LFE scheme by interleaving the circuits C'_i in our construction above (which essentially perform read queries on the receiver's database) with the circuits C''_i originating from an underlying garbled RAM.

As a result, we get a fully-fledged RAM LFE scheme with ciphertext size, sender, and receiver overheads of $O_\lambda(T \cdot \sqrt{n})$. By relying on a black-box garbled RAM scheme [GLO15,HKO22] and storing the *expanded laconic OT correlations*, the garbled circuits need only perform black-box operations. Consequently, for moderately large parameters n, the overhead of our scheme compares favorably to the RAM-LFE scheme in [DHMW24], which relies on heavy non-black-box techniques.

8 Laconic Private Set Intersection with Preprocessing

In a laconic private set intersection (PSI) protocol [ABD+21, ALOS22, DKL+23], a receiver holding a potentially huge set X of size n and a sender holding a small set Y of size m can jointly compute the intersection $X \cap Y$ without revealing any other information about their sets to each other. In previous works, they all require a linear computation complexity on the receiver's side. We now use our private laconic OT with preprocessing scheme which guarantees receiver privacy to construct a preprocessed laconic PSI protocol where the receiver computation complexity is consistent with our private laconic OT scheme.

Construction 5 (Laconic PSI with Preprocessing) *Let* $h : \{0,1\}^* \mapsto \{0,1\}^n$ *be a collision-resistent hash function and* $\mathsf{IOT} = (\mathsf{Setup}, \mathsf{Offline}, \mathsf{Send}, \mathsf{Receive})$ *be a private lOT with preprocessing scheme that guarantees receiver privacy. The scheme* $\mathsf{LPSI} = (\mathsf{LPSI.Setup}, \mathsf{LPSI.Offline}, \mathsf{LPSI.Send}, \mathsf{LPSI.Receive})$ *is given as follows.*

- $\mathsf{pp} \leftarrow \mathsf{LPSI.Setup}(1^\lambda)$*:*
  1. *Compute and output* $\mathsf{pp} \leftarrow \mathsf{IOT.Setup}(1^\lambda)$.
- $\mathsf{cor} \leftarrow \mathsf{LPSI.Offline}(\mathsf{pp}, X)$*:*
  1. *Initialize a database* DB *of length* n*, and for* $x_j \in X$*, set* $\mathsf{DB}_{h(x_j)} = 1$*;*

   2. *Compute and output* $\mathsf{cor} \leftarrow \mathsf{IOT.Offline}(\mathsf{pp}, \mathsf{DB})$.
- $\mathsf{msg} \leftarrow \mathsf{LPSI.Send}(\mathsf{pp}, \mathsf{cor}, Y)$*:*
   1. *For all* $y_j \in Y$*, generate two messages* $m_{j,0}, m_{j,1}$ *as follows: set a message* $m_{j,1} = 0^\lambda \| y_j$ *and sample a randomness* $m_{j,0}$ *of the same length with the first bit being* 1*, and then compute* $(e_j, l) \leftarrow \mathsf{IOT.Send}(\mathsf{pp}, \mathsf{cor}, h(y_j), m_{j,0}, m_{j,1})$*;*
   2. *Pick a permutation* $\pi : [\mathsf{m}] \mapsto [\mathsf{m}]$*;*
   3. *Output* $\mathsf{msg} = \{e_{\pi(j)}\}_{j \in [\mathsf{m}]}$.
- $c \leftarrow \mathsf{LPSI.Receive}(\mathsf{pp}, \mathsf{DB}, \mathsf{msg})$*:*
   1. *Parse* $\mathsf{msg} = \{e_j\}_{j \in [\mathsf{m}]}$ *and initialize an empty intersection set* Z*;*
   2. *For* $j \in [\mathsf{m}]$*, compute* $m_j = \mathsf{IOT.Receive}(\mathsf{pp}, \mathsf{DB}, e_i)$*. If* m_j *is of form* $0^\lambda \| z$*, add* z *into the set* Z.
   3. *Output* Z.

Correctness. For simplicity, we first assume the sender holds an element y and $y \in X$. Upon receiving e, the receiver will output y since $0^\lambda \| y = m_{\mathsf{DB}_{h(y)}} \leftarrow \mathsf{IOT.Receive}(\mathsf{pp}, \mathsf{DB}, e)$ where $\mathsf{DB}_{h(y)} = 1$. To get the correctness of the entire protocol, we repeat this procedure for each element in the sender's set.

Efficiency. The efficiency of our laconic PSI with preprocessing protocol aligns with that of the underlying private laconic OT protocol. Specifically, the offline communication takes $\tilde{O}_\lambda(\sqrt{n})$, the online communication takes $\tilde{O}_\lambda(m \cdot \sqrt{n})$. During the online phase, the sender and the online server both run in time $O_\lambda(m \cdot \sqrt{n})$.

Security. The following theorem establishes security for our laconic PSI with preprocessing.

Theorem 4 *If* IOT *is a private laconic OT with preprocessing scheme and provides receiver privacy and the hash function* h *is collision-resistant, the laconic PSI with preprocessing protocol shown in Construction 5 is secure in the semi-honest model.*

Proof. Intuitively, the security against corrupted sender is from the receiver privacy, i.e., the correlations cor hide the indices of non-zero entries with respect to the receiver's private input. The security against corrupted receiver is from the security property of the encryption scheme such that a ciphertext with respect to an index not registered looks pseudorandom. We now show that it is possible to simulate the transcripts for both parties which are computationally indistinguishable from the view in a real world.

Security Against Corrupted Sender. On input the set S_S and nothing else, we simulate the view of the corrupted sender by running the simulator for receiver privacy of the underlying private laconic OT protocol which outputs an indistinguishable transcript as in the real world.

Security Against Corrupted Receiver. On input the set S_R and the intersection set Z, we simulate the view of a corrupted receiver in the following way:

1. Simulate the setup phase as an honest party, i.e., $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. Pick a random subset $\Gamma \subset [\mathsf{m}]$ with $|\Gamma| = \zeta$. Let $\Gamma = \{\gamma_1, \ldots, \gamma_\zeta\}$. For all $j \in [\zeta]$, set a message $m_{\gamma_j} = 0^\lambda \| z_j$ and invoke the simulator for sender privacy of the underlying laconic OT with preprocessing scheme to compute $e_{\gamma_j} \leftarrow \mathsf{IOT.Sim}(\mathsf{pp}, \mathsf{DB}, m_{\gamma_j})$. This, $(e_{\gamma_j})_{j \in [\zeta]}$, simulates the positions where the receiver gets a match.
3. Let the complimentary set be $\Delta = [\mathsf{m}] \setminus Z = \{\delta_1, \ldots, \delta_\eta\}$ where $\eta = \mathsf{m} - \zeta$. For all $j \in [\eta]$, sample a pair of randomness $r_{\delta_j}, r'_{\delta_j} \leftarrow\!\!\$\ \{0,1\}^\lambda$, a vector $\mathbf{v} \leftarrow\!\!\$\ \mathbb{F}_q^n$ constrained by $\mathsf{hw}(\mathbf{v}) = \sqrt{n}$ and $\sum_{i \in [n]} v_i \cdot \mathsf{DB}_i = 0$. and compute $(e_{\eta_j}, l_j) \leftarrow \mathsf{IOT.Send}\left(\mathsf{pp}, \langle \mathbf{v}, \mathsf{DB} \rangle, h(w), r'_{\delta_j}, r_{\delta_j}\right)$ with a random $w \notin Z$. This, $(e_{\delta_j})_{j \in [\eta]}$, simulates the positions where the receiver does not get a match.
4. Output $\{e_j\}_{j \in [\mathsf{m}]}$ as the simulated view.

We now show that the simulated view is indistinguishable from the view in the real execution using the following sequence of hybrids.

- Hyb_0: Identical to the view of the receiver in the real protocol.

- $\mathsf{Hyb}_{1,0}$: Instead of sampling the permutation π, we pick sets (Γ, Δ) where $\Gamma = \{\gamma_1, \ldots, \gamma_\zeta\}$ and Δ is the complimentary set, like in the simulation. For each j we find the index $\sigma(j)$ such that $z_j = y_{\sigma(j)}$ and choose a random permutation π such that $\pi(\gamma_j) = \sigma(j)$ and the remaining positions are filled at random.
  The indistinguishability between Hyb_0 to $\mathsf{Hyb}_{1,0}$ is immediate since in both cases π is a uniform permutation in $[\mathsf{m}]$.
- $\mathsf{Hyb}_{1,j}$ for $j \in [n]$: In each hybrid, we change the distribution of a single e_j such that in $\mathsf{Hyb}_{1,j-1}$ it is generated as in the real protocol while in $\mathsf{Hyb}_{1,j}$ it is generated as in the simulation.
  For $j \in \Gamma$, e_j is indistinguishable from that in the real protocol given the sender privacy of the private laconic OT protocol, and thus $\mathsf{Hyb}_{1,j-1}$ and $\mathsf{Hyb}_{1,j}$ for $j \in \Gamma$ is indistinguishable.
  For $j \in \Delta$, since when there is not a match, there is no correlation that holds. Given that the non-zero entries on the vector are sampled randomly, the keys for the IND-CPA secure symmetric encryption in the $\mathsf{IOT.Send}$ algorithm are just random and in the real protocol the ciphertext is pseudorandom. Therefore e_j is indistinguishable from that in the real protocol, and thus $\mathsf{Hyb}_{1,j-1}$ and $\mathsf{Hyb}_{1,j}$ for $j \in \Delta$ is indistinguishable.
- Hyb_2: Identical to the view of R in the simulation.
  In $\mathsf{Hyb}_{1,n}$, each e_j from the real protocol has been replaced with that from the simulation. Therefore, $\mathsf{Hyb}_{1,n}$ and Hyb_2 are indistinguishable, and this concludes the proof.

□

Acknowledgments

We thank Lisa Kohl and Stella Wohnig for discussing index privacy in our construction.

Nico Döttling and Chuanwei Lin are funded by the European Union (ERC, LACONIC, 101041207). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

Carmit Hazay is supported by the United States-Israel Binational Science Foundation (BSF) through Grant No. 2020277.

References

ABD+21. N. Alamati, P. Branco, N. Döttling, S. Garg, M. Hajiabadi, and S. Pu. Laconic private set intersection and applications. In *TCC 2021, Part III*, *LNCS* 13044, pages 94–125. Springer, Cham, November 2021.

ALOS22. D. F. Aranha, C. Lin, C. Orlandi, and M. Simkin. Laconic private set-intersection from pairings. In *ACM CCS 2022*, pages 111–124. ACM Press, November 2022.

BCC+16. J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *EUROCRYPT 2016, Part II*, *LNCS* 9666, pages 327–357. Springer, Berlin, Heidelberg, May 2016.

BCG+19a. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, P. Rindal, and P. Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In *ACM CCS 2019*, pages 291–308. ACM Press, November 2019.

BCG+19b. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *CRYPTO 2019, Part III*, *LNCS* 11694, pages 489–518. Springer, Cham, August 2019.

BCG+22. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, N. Resch, and P. Scholl. Correlated pseudorandomness from expand-accumulate codes. In *CRYPTO 2022, Part II*, *LNCS* 13508, pages 603–633. Springer, Cham, August 2022.

BCG+23. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, N. Resch, and P. Scholl. Oblivious transfer with constant computational overhead. In *EUROCRYPT 2023, Part I*, *LNCS* 14004, pages 271–302. Springer, Cham, April 2023.

BCW03. G. Brassard, C. Crépeau, and S. Wolf. Oblivious transfers and privacy amplification. *Journal of Cryptology*, 16(4):219–237, September 2003.

BDOZ11. R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias. Semi-homomorphic encryption and multiparty computation. In *EUROCRYPT 2011*, *LNCS* 6632, pages 169–188. Springer, Berlin, Heidelberg, May 2011.

Bea92. D. Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO'91*, *LNCS* 576, pages 420–432. Springer, Berlin, Heidelberg, August 1992.

Bea96. D. Beaver. Correlated pseudorandomness and the complexity of private computations. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 479–488. ACM, 1996.

BGI19. E. Boyle, N. Gilboa, and Y. Ishai. Secure computation with preprocessing via function secret sharing. In *TCC 2019, Part I*, *LNCS* 11891, pages 341–371. Springer, Cham, December 2019.

BGKW88. M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *20th ACM STOC*, pages 113–131. ACM Press, May 1988.

BLN+21. S. S. Burra, E. Larraia, J. B. Nielsen, P. S. Nordholt, C. Orlandi, E. Orsini, P. Scholl, and N. P. Smart. High-performance multi-party computation for binary circuits based on oblivious transfer. *Journal of Cryptology*, 34(3):34, July 2021.

BV11a. Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106. IEEE Computer Society, 2011.

BV11b. Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *CRYPTO*, pages 505–524, 2011.

CDE+18. R. Cramer, I. Damgård, D. Escudero, P. Scholl, and C. Xing. SPD $\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. In *CRYPTO 2018, Part II*, *LNCS* 10992, pages 769–798. Springer, Cham, August 2018.

CDG+17. C. Cho, N. Döttling, S. Garg, D. Gupta, P. Miao, and A. Polychroniadou. Laconic oblivious transfer and its applications. In *CRYPTO 2017, Part II*, *LNCS* 10402, pages 33–65. Springer, Cham, August 2017.

CK20. H. Corrigan-Gibbs and D. Kogan. Private information retrieval with sublinear online time. In *EUROCRYPT 2020, Part I*, *LNCS* 12105, pages 44–75. Springer, Cham, May 2020.

Cou19. G. Couteau. A note on the communication complexity of multiparty computation in the correlated randomness model. In *EUROCRYPT 2019, Part II*, *LNCS* 11477, pages 473–503. Springer, Cham, May 2019.

DGI+19. N. Döttling, S. Garg, Y. Ishai, G. Malavolta, T. Mour, and R. Ostrovsky. Trapdoor hash functions and their applications. In *CRYPTO 2019, Part III*, *LNCS* 11694, pages 3–32. Springer, Cham, August 2019.

DHM+24. F. Dong, Z. Hao, E. Mook, H. Wee, and D. Wichs. Laconic function evaluation and ABE for RAMs from (ring-)LWE. In *CRYPTO 2024, Part III*, *LNCS* 14922, pages 107–142. Springer, Cham, August 2024.

DHMW24. F. Dong, Z. Hao, E. Mook, and D. Wichs. Laconic function evaluation, functional encryption and obfuscation for RAMs with sublinear computation. In *EUROCRYPT 2024, Part II*, *LNCS* 14652, pages 190–218. Springer, Cham, May 2024.

DKL+13. I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In *ESORICS 2013*, *LNCS* 8134, pages 1–18. Springer, Berlin, Heidelberg, September 2013.

DKL+23. N. Döttling, D. Kolonelos, R. W. F. Lai, C. Lin, G. Malavolta, and A. Rahimi. Efficient laconic cryptography from learning with errors. In *EUROCRYPT 2023, Part III*, *LNCS* 14006, pages 417–446. Springer, Cham, April 2023.

DNNR17. I. Damgård, J. B. Nielsen, M. Nielsen, and S. Ranellucci. The TinyTable protocol for 2-party secure computation, or: Gate-scrambling revisited. In *CRYPTO 2017, Part I*, *LNCS* 10401, pages 167–187. Springer, Cham, August 2017.

DPSZ12. I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO 2012*, *LNCS* 7417, pages 643–662. Springer, Berlin, Heidelberg, August 2012.

DZ13. I. Damgård and S. Zakarias. Constant-overhead secure computation of Boolean circuits using preprocessing. In *TCC 2013*, *LNCS* 7785, pages 621–641. Springer, Berlin, Heidelberg, March 2013.

FHAS24. N. Fleischhacker, M. Hall-Andersen, and M. Simkin. Extractable witness encryption for kzg commitments and efficient laconic ot. In *ASIACRYPT*. Springer-Verlag, 2024.

Gen09. C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

GHL+14. C. Gentry, S. Halevi, S. Lu, R. Ostrovsky, M. Raykova, and D. Wichs. Garbled RAM revisited. In *EUROCRYPT 2014*, *LNCS* 8441, pages 405–422. Springer, Berlin, Heidelberg, May 2014.

GHMM18. S. Garg, M. Hajiabadi, M. Mahmoody, and A. Mohammed. Limits on the power of garbling techniques for public-key encryption. In *CRYPTO 2018, Part III*, *LNCS* 10993, pages 335–364. Springer, Cham, August 2018.

GJL23. M. Green, A. Jain, and G. V. Laer. Efficient set membership encryption and applications. In *ACM CCS 2023*, pages 1080–1092. ACM Press, November 2023.

GLO15. S. Garg, S. Lu, and R. Ostrovsky. Black-box garbled RAM. In *56th FOCS*, pages 210–229. IEEE Computer Society Press, October 2015.

GMW87. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

GSW13. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, pages 75–92. Springer, 2013.

HKO22. D. Heath, V. Kolesnikov, and R. Ostrovsky. EpiGRAM: Practical garbled RAM. In *EUROCRYPT 2022, Part I*, *LNCS* 13275, pages 3–33. Springer, Cham, May / June 2022.

HOSS18. C. Hazay, E. Orsini, P. Scholl, and E. Soria-Vazquez. Concretely efficient large-scale MPC with active security (or, TinyKeys for TinyOT). In *ASIACRYPT 2018, Part III*, *LNCS* 11274, pages 86–117. Springer, Cham, December 2018.

IKM+13. Y. Ishai, E. Kushilevitz, S. Meldgaard, C. Orlandi, and A. Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In *TCC 2013*, *LNCS* 7785, pages 600–620. Springer, Berlin, Heidelberg, March 2013.

IKNP03. Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In *CRYPTO 2003*, *LNCS* 2729, pages 145–161. Springer, Berlin, Heidelberg, August 2003.

IR90. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *CRYPTO'88*, *LNCS* 403, pages 8–26. Springer, New York, August 1990.

Kil88a. J. Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988.

Kil88b. J. Kilian. Zero-knowledge with log-space verifiers. In *29th FOCS*, pages 25–35. IEEE Computer Society Press, October 1988.

LMW23. W. Lin, E. Mook, and D. Wichs. Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE. In *STOC*, pages 595–608. ACM, 2023.

LO13. S. Lu and R. Ostrovsky. How to garble RAM programs. In *EUROCRYPT 2013*, *LNCS* 7881, pages 719–734. Springer, Berlin, Heidelberg, May 2013.

NNOB12. J. B. Nielsen, P. S. Nordholt, C. Orlandi, and S. S. Burra. A new approach to practical active-secure two-party computation. In *CRYPTO 2012*, *LNCS* 7417, pages 681–700. Springer, Berlin, Heidelberg, August 2012.

PAP24. E. Pohle, A. Abidin, and B. Preneel. Fast evaluation of s-boxes with garbled circuits. *IEEE Trans. Inf. Forensics Secur.*, 19:5530–5544, 2024.

QWW18. W. Quach, H. Wee, and D. Wichs. Laconic function evaluation and applications. In *59th FOCS*, pages 859–870. IEEE Computer Society Press, October 2018.

Rab81. M. O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University, 1981.

Reg05. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

Yao82. A. C.-C. Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.

YWL+20. K. Yang, C. Weng, X. Lan, J. Zhang, and X. Wang. Ferret: Fast extension for correlated OT with small communication. In *ACM CCS 2020*, pages 1607–1626. ACM Press, November 2020.

YWZ20. K. Yang, X. Wang, and J. Zhang. More efficient MPC from improved triple generation and authenticated garbling. In *ACM CCS 2020*, pages 1627–1646. ACM Press, November 2020.