Credibility in Private Set Membership

Sanjam Garg[1,2], Mohammad Hajiabadi[3], Abhishek Jain[4], Zhengzhong Jin[5], Omkant Pandey[6], and Sina Shiehian[7]

[1] University of California, Berkeley
[2] NTT Research
[3] University of Waterloo
[4] Johns Hopkins University
[5] MIT
[6] Stony Brook University
[7] Snap Inc.

Abstract. A private set membership (PSM) protocol allows a "receiver" to learn whether its input x is contained in a large database DB held by a "sender". In this work, we define and construct *credible private set membership (C-PSM)* protocols: in addition to the conventional notions of privacy, C-PSM provides a soundness guarantee that it is hard for a sender (that does not know x) to convince the receiver that $x \in \mathsf{DB}$. Furthermore, the communication complexity must be logarithmic in the size of DB.

We provide 2-round (i.e., round-optimal) C-PSM constructions based on standard assumptions:

– We present a black-box construction in the plain model based on DDH or LWE.
– Next, we consider protocols that support predicates f beyond string equality, i.e., the receiver can learn if there exists $w \in \mathsf{DB}$ such that $f(x, w) = 1$. We present two results with transparent setups: (1) A black-box protocol, based on DDH or LWE, for the class of NC^1 functions f which are efficiently searchable. (2) An LWE-based construction for all bounded-depth circuits. The only non-black-box use of cryptography in this construction is through the bootstrapping procedure in fully homomorphic encryption.

As an application, our protocols can be used to build enhanced round-optimal leaked password notification services, where unlike existing solutions, a dubious sender *cannot* fool a receiver into changing its password.

1 Introduction

A two-party private set membership (PSM) protocol is an interactive protocol between a receiver holding an input x and a sender holding a database DB. The goal is that at the end of the interaction, the receiver only learns whether $x \in \mathsf{DB}$ while the sender learns nothing about x. Similar to private information retrieval [8], a desirable feature for PSM is efficiency of the receiver, which states that the communication complexity and also the computational complexity of the receiver is sublinear (or more preferably logarithmic) in the size of DB. PSM and its closely related variant private set intersection (PSI) have found numerous applications such as contact discovery [17] and exposed password notification [11,16,2].

In the exposed password notification use-case, a user and a service provider run a PSM protocol to determine whether the user's password is exposed in any leaked database. An often neglected aspect in this setting is whether the protocol provides a *credible* guarantee to the user that its password was actually leaked. In fact, a dubious sender might potentially keep falsely suggesting to the user that its password was exposed, causing the user to go through the process of updating its password.

A potential approach to enforce credibility might be requiring the sender to send its whole database in an encrypted format. It is plausible that such an approach, specially when implemented through protocols based on oblivious pseudorandom functions (OPRF) [11,2], can provide credibility. However, sending the whole database would obviously make the protocol's communication and the receiver's computational complexity linear in the size of the database, and thus violates efficiency. Another approach may be using generic cryptographic succinct zero-knowledge arguments of knowledge. Such solutions incur an unsatisfactory computational overhead due to the use of *non-black-box* techniques. Therefore, we ask

Can we construct asymptotically efficient black-box credible PSM protocols?

1.1 Our Contributions

Defining C-PSM. In this work we initiate the study of *credibility* in PSM protocols. We define the notion of *credible private set membership* (C-PSM). Informally, a C-PSM for a relation $\mathcal{R}$ is a two party protocol between a receiver and a sender where both the receiver and the sender have access to a common reference string (CRS). The receiver has an input x and the sender has a large database DB. The sender wants to convince the receiver that the database contains a witness w such that $(x, w) \in \mathcal{R}$. We require the following properties:

- The protocol consists of only two rounds.
- The communication and also the receiver's computational complexity is at most logarithmic in the size of DB.
- The receiver's input x remains hidden from the sender.
- The sender's database remains private, i.e., a (malicious) receiver does not learn anything more than the fact that the database contains a valid witness.
- The protocol is sound, i.e., if the sender does not have a witness in the database, then, it is computationally hard for it to make the receiver accept.

We focus on black-box protocols, i.e., protocols which only make black-box use of their underlying cryptographic tools. For the soundness property to be meaningful and achievable in 2 rounds, we require the input x to have high entropy. Otherwise, if x is predictable, the sender can always include a valid witness for x in its database and convince the receiver. For the same reason we consider relations $\mathcal{R}$ which are *instance entropic*. Roughly speaking, this means that any witness only satisfies a negligible fraction of instances. For example, the string equality relation is instance entropic.

C-PSM for String Equality. We start by considering the basic string equality relation, where the receiver wants to check if $x \in \mathsf{DB}$. For this relation we construct a black-box 2-round C-PSM protocol in the plain model from either of the DDH or LWE assumption.

Theorem 1 (Informal). *Assuming the hardness of either of DDH or LWE, there exists a black-box 2-round C-PSM protocol in the plain model for the string equality relation.*

C-PSM for Efficiently Searchable Relations. We then turn to instance entropic relations beyond string equality. Specifically, we will consider the scenario where for some function f, the receiver wants to check whether DB contains c entries $w_1, \cdots, w_c$ such that $f(x, \{w_i\}_{i \in [c]}) = 1$. We first consider the class of *efficiently searchable* functions, i.e., functions which are in NC^1, and, for any input x, searching DB for witnesses can be implemented by a branching program of length logarithmic in DB. We construct a fully black-box 2-round C-PSM protocol for the class of *efficiently searchable* functions assuming either of DDH or LWE.

Theorem 2 (Informal). *Assuming the hardness of either of DDH or LWE, for every searchable function there exists a black-box 2-round C-PSM protocol with transparent setup.*

Next, we construct a C-PSM from LWE which is not restricted to efficiently searchable functions and supports all bounded-depth circuits. While this construction is not fully black-box, however, its non-black-use of cryptography is limited to the bootstrapping procedure in its underlying homomorphic encryption.

Theorem 3 (Informal). *Assuming the hardness of LWE, there exists a 2-round C-PSM protocol with transparent setup for every (bounded-depth) circuit. The only non-black-box use of cryptography in this C-PSM protocol is through bootstrapping in homomorphic encryption.*

We mention that all of our C-PSM protocols satisfy *statistical sender privacy*. This means that, our constructions guarantee the privacy of the sender even against computationally unbounded malicious receivers. Additionally, in our constructions which need a setup, receiver privacy is guaranteed even if the CRS is maliciously generated.

Applications. Our construction for string equality immediately gives a credible protocol for password exposure notification. In fact, since the C-PSM protocol in this construction only consists of two rounds, the receiver can publish its first message and wait for senders to inform him/her of a password exposure via C-PSM second message.

With our black-box construction for efficiently searchable relations, we can have protocols that perform more complicated tasks. For instance, consider a situation where the sender's database consists of pairs of usernames and candidate passwords. A receiver wants to learn whether the database has an entry consisting of its username paired with a closely matching password (closely matching can for example mean having an edit distance no bigger than half the length of the password). We observe that our black-box construction supports this functionality. This is because given a username and password pair, the following branching program whose length is logarithmic in the size of the database can implement the corresponding search functionality:

1. First, search the database for an entry with a matching username. Note that this step can be implemented by a logarithmic length branching program through using the trie data structure.
2. Next, given an entry with a matching username, check whether the candidate password in the entry closely matches the input password. This step is independent of the size of the database and can be implemented by an NC^1 circuit, and consequently by a polynomial sized branching program.

1.2 Related Work

The notion of zero-knowledge sets [19] allows a sender to convince a receiver whether an element exist in its database or not by sending a short proof. Our work differs from zero-knowledge sets in two aspects. First, we consider 2-round protocols whereas zero-knowledge sets consist of protocols having 3 rounds, where, in the first round the sender commits to its database and publishes a digest of this commitments. Second, there is no receiver privacy in zero-knowledge sets, i.e., the receiver sends its input in the clear.

A line of work [7,6,9] constructed concretely efficient *unbalanced* PSI protocols, i.e., PSI protocols where the sender's set is considerably larger than the receiver's set, from FHE. The PSI protocols constructed in these works provide sender privacy, receiver privacy and and communication sub-linear in the size of the sender's set. While exposed password notification seems to be one of the main applications of the PSI protocols constructed in these works, however, they do not provide credibility. In fact [6] considers a heuristic approach to make it more difficult for a dubious sender to cheat. Roughly speaking, the proposal in [6] requires a sender to include the hash of the receiver's input in the FHE ciphertext that it outputs. Then, it sets the FHE parameters such that it does not support computing this hash function. Our construction for string equality in section 4 can be seen as a dual of this idea, where, we use the output of a one-way function as the input and treat the original input as the *label*. Unlike [6], we are able to formally prove the credibility of our construction.

Another work [15] considers oblivious polynomial evaluation (OPE). In this setting, the receiver wants to learn the image of its private input under a secret high-degree polynomial that is held by the sender. Notice that an instance of PSM can be converted into an instance of OPE where the degree of the polynomial is equal to the size of the database. The protocol in [15] provides receiver privacy, sender privacy and communication sub-linear in the degree of the polynomial. Additionally, this construction ensures that the evaluated value that receiver obtains truly corresponds to the polynomial that is held by the sender. While the latter property can be viewed as credibility, however, the way [15] enforces this property is by requiring the sender to send a commitment to its polynomial to the receiver. Consequently, this protocol needs three rounds.

1.3 Acknowledgments

Sanjam Garg is supported in part by DARPA under Agreement No. HR00112020026, AFOSR Award FA9550-19-1-0200, NSF CNS Award 1936826, and research grants by the Sloan Foundation, and Visa Inc. Omkant Pandey is supported in part by DARPA SIEVE Award HR00112020026, NSF CAREER Award 2144303, NSF grants 2028920, 2106263, and 2128187. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government, DARPA, Sloan Foundation, Visa Inc., or NSF.

1.4 Techniques

C-PSM for string equality. We start by providing an overview of our C-PSM construction for the basic string equality functionality. Since we are aiming to keep the receiver's complexity independent of the size of the database, it is natural to consider using homomorphic encryption (HE). However, a naive scheme where the receiver sends its input x encrypted under FHE, and the sender homomorphically searches its database, does not satisfy the properties of C-PSM:

- First and foremost, this construction is not credible because the sender can simply send a homomorphically encrypted positive answer regardless of its database.
- Furthermore, this construction does not provide sender privacy because homomorphic evaluation might reveal extra information about the sender's database.

Our insight to solve the first issue is noticing that the receiver's input has high entropy and therefore it is hard to invert its image under a one-way function. Specifically, the receiver, instead of sending an encryption of its input, sends an encryption of the image $y = f(x)$ of its input under a one-way function f. The sender computes the images of all entries in its database under f and proceeds to homomorphically search these images for y. If found, the sender can homomorphically include the pre-image x in the ciphertext it sends to the receiver.

To add sender privacy, we will use a homomorphic encryption scheme with a property known in the literature as *malicious function privacy* [21]. Informally, this notion states that the evaluated ciphertexts reveal nothing beyond the value they are encrypting, and in particular they hide the function that was homomorphically evaluated. While the malicious function private HE construction in [21] makes extensive non-black-box use of cryptography, however, fortunately, we can instantiate the OT-based black-box HE construction in [14] with the recent rate-1 statistical sender private OT [1], which can be based on either LWE or DDH, to get a black-box malicious function private HE for branching programs.

Beyond string equality. We now describe how we build a C-PSM supporting predicates beyond string equality. For the ease of exposition, we present a 4-round protocol and then briefly sketch how we compress it to 2 rounds. Recall that in this setting, the receiver holds an input x and the sender wants to convince the receiver that its database contains a witness w such that $f(x, w) = 1$ for a specific predicate f. Our starting idea is to use homomorphic encryption for encrypting the the receiver's input, a black-box commit-and-prove system for committing to the sender's database and generating zero-knowledge proofs, and Merkle trees [18] for creating a digest of this database. In more detail, similar to the string quality construction, the receiver encrypts its input under HE and sends the ciphertext to the sender. The sender then works as follows:

- First, it commits to the database using the commit and prove system, i.e., it secret shares each entry in the database and commits to these shares.
- Next, it hashes these commitments using a Merkle tree.
- Then, it homomorphically searches the database to find a valid witness w along with a Merkle hash opening for its corresponding commitment (or $\perp$ if the database does not contain such a witness). Note that this does not involve any hash computations under the hood of HE. All hashes can be computed "outside," and then moved to under the hood of HE.
- Next, the sender homomorphically generates the first prover message in the commit-and-prove system and sends it to the receiver.
- Finally, upon receiving a challenge from the receiver, the sender homomorphically opens a subset of the commitments produced in the first message and sends them to the receiver.

While this approach has succinct communication complexity, keeps the receiver's input private, and is black-box thanks to the MPC-in-the-head [13] paradigm, however, it fails to protect against a malicious sender. In fact, a malicious sender whose database does not contain a valid witness can homomorphically cook up a database containing a witness and proceed to deceive the receiver. A straightforward approach to provide security against malicious senders is to require the sender to attach (in plain) a succinct non-interactive argument of knowledge (SNARK), showing that the evaluated ciphertext is the result of an honest evaluation using an actual database known by the sender. However, in addition to relying on unfalsifiable assumptions, this approach results in a very prohibitive solution and involves expensive non-black-box use of cryptography. For string equality we were able to overcome this issue by using deterministic encryption, but for richer functionalities this idea does not seem to be applicable. In summary, with the goal of avoiding expensive cryptography, the main challenge we face is "how do we tie the hands of a malicious sender to prevent it from cooking up a database under the hood of homomorphic encryption?"

First Attempt: Attaching the hash root "outside." Our first starting idea for tying the hands of the malicious sender is to have it send something "outside" the homomorphic encryption wrapper. The sender could cook up stuff under homomorphic encryption but cannot do so outside! The receiver could then compare the information obtained under the hood of HE and check if it is consistent with the information provided "outside." The hope is that given that a malicious sender cannot cook up stuff depending on receiver's input "outside," consistency is only possible if a valid witness exists in the database.

In particular, if we require the sender to include the root of the Merkle tree *in clear*, then, the homomorphic database cooking up attack that we described in the previous paragraph does not seem to work. Intuitively, the hash root seems to *bind* the prover to a database in clear, and if this database (and consequently the hash root) depends on the receiver's input, then, a cheating prover has to somehow break the security of HE.

However, unfortunately, it is unclear how to prove security of this strategy. In other words, it is unclear how we could reduce the ability of the sender to break soundness to breaking the security of HE or the Merkle hash. A key issue is that the hash root does not have any *extractable information* to help with breaking the security of HE.

Using SSB hashing to make a random point extractable. In order to fix the above issue, while avoiding expensive tools, we try for a very simple approach. In particular, we replace the generic Merkle Hash with a somewhere statistically binding (SSB) hash [12]. At a high level, SSB hash is a Merkle tree with an additional binding property. In more detail, in a SSB hash, the hashing key can be generated for binding to a specific position i in the input. The guarantee is that, the hash root now *statistically binds* to commitments to the value of the database at position i, which remains computationally hidden by the *index hiding* property. We assume a stronger *extractability* guarantee from our SSB hash. Namely, we assume that it is possible to *extract* the ith value given only the hash root and a *extraction trapdoor* which is generated along with the hashing key. Fortunately, these objects can be built based on any rate-1 OT using previous known techniques [12,20].

Somewhat surprising, though with a subtle argument, this simple change allows us to reduce a malicious sender's ability to cheat to break the security of HE or violate the index-hiding property of the SSB hash. We now sketch how using extractable SSB hashing we can reduce the security of HE to the soundness of C-PSM. Our reduction simply generates a SSB hash key binding to a *uniformally random* position and puts it in the CRS. First, observe that the index hiding property of SSB hash ensures that, during the execution, with noticeable probability, this random position is the same position that the cheating sender opens under the hood of HE. Clearly, if the adversary can somehow always avoid the random position encoded in the SSB hash key then that adversary can be used to break the index hiding property of SSB with probability better than a random guess. In the final step, we show a reduction that uses the value extracted from the SSB hash root — which from the prior step we know is correlated with the encrypted value under HE with a small probability — to directly break the security of HE.

Instantiating HE. Similar to our construction for string equality, we can use the malicious circuit private HE for branching programs that can be instantiated by combining [1] and [14]. For achieving compact communication complexity when using this instantiation of HE, searching the database for a witness should be implementable with a branching program whose length is logarithmic in the size of the database. That is, the predicate should be *efficiently searchable*. This is because in the [14] HE construction, the size of evaluated ciphertexts grow linearly in the length of the evaluated branching programs.

Another option is to use the LWE-based malicious circuit private HE in [10]. With this HE, our C-PSM construction can support every instance entropic predicate that can be implemented by a (bounded-depth) circuit. However, the HE in [10] is not fully black-box as it performs bootstrapping for every evaluation.

Black-box commitment generation. A delicate issue is that, the sender algorithm, as currently described, would be non-black-box, because, generating the first prover message for the commit-and-prove system involves generating new commitments. We avoid this non-black-box step via the following trick: the sender generates many fresh commitments to 0 and 1 in the clear and then, obliviously brings these fresh commitments under HE based on the message the prover commits to.

4-Round to 2-round. Finally, we describe how to compress the described 4-round C-PSM to a 2-round protocol. To do this, the receiver sends its challenge via OT in the first round. In the second round, the sender prepares a C-PSM sender's message for each possible challenge and sends them to the receiver through OT response.

2 Preliminaries

We denote the security parameter by λ. For any $\ell \in \mathbb{N}$, we denote the set of the first ℓ positive integers by $[\ell]$. For a set S, $x \leftarrow S$ denotes sampling a uniformly random element x from S. For a distribution D, $x \leftarrow D$ denotes sampling an element x from D.

2.1 Oblivious Transfer

We review the definition of rate-1 statistical sender private oblivious transfer.

Definition 1 (Rate-1 Statistical Sender Private Oblivious Transfer). *A (string) 1-out-of-2 OT consists of three algorithms:* $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$.

- $\mathsf{OT}_1(1^\lambda, b)$, *on input a security parameter* $\lambda \in \mathbb{N}$ *and a choice bit* $b \in \{0,1\}$, *outputs a protocol message* ot_1 *and a state* st.
- $\mathsf{OT}_2(ot_1, (m_0, m_1))$, *on input* ot_1, *and two sender inputs* (m_0, m_1) *of the same length, outputs a response* ot_2.
- $\mathsf{OT}_3(st, ot_2)$, *on input a state* st *and* ot_2, *outputs a message* m.

We require the following properties:

1. Correctness, *for all security parameters* λ, *bits* $b \in \{0,1\}$, *and sender inputs* $m_0, m_1 \in \{0,1\}^*$:

$$\Pr\left[y = m_b \;\middle|\; \begin{array}{l} (ot_1, st) \leftarrow \mathsf{OT}_1(1^\lambda, b) \\ ot_2 \leftarrow \mathsf{OT}_2(ot_1, (m_0, m_1)) \\ y \leftarrow \mathsf{OT}_3(st, ot_2) \end{array} \right] = 1.$$

2. Receiver Security, $ot \overset{c}{\approx} ot'$, *where* $(ot, *) \leftarrow \mathsf{OT}_1(1^\lambda, 0)$ *and* $(ot', *) \leftarrow \mathsf{OT}_1(1^\lambda, 1)$.
3. Statistical Sender Privacy, *there exists an unbounded simulator* $\mathcal{S}$ *such that for all (not necessarily honestly generated)* ot_1 *there exists a bit* b, *such that for all sender inputs* $m_0, m_1 \in \{0,1\}^*$:

$$\mathsf{OT}_2(ot_1, (m_0, m_1)) \overset{s}{\approx} \mathsf{Sim}(1^\lambda, ot_1, m_b)).$$

4. Rate-1: *There exists a fixed polynomial* poly *such that for all polynomials* $n := n(\lambda)$, *for all first-round messages* ot_1 *and for all* $(m_0, m_1) \in \{0,1\}^n \times \{0,1\}^n$, $|ot_2| = n + \mathsf{poly}(\lambda)$, *where* $ot_2 \leftarrow \mathsf{OT}_2(ot_1, (m_0, m_1))$.

Theorem 4 ([1]). *Assuming either DDH or LWE, there exists a black-box construction of rate-1 statistical sender private OT.*

We also consider the following dual-mode variation of OT. Notice that this variation is not rate-1.

Definition 2 (Dual-mode OT). *Let* C *be a constant. A* 1*-out-of-*C *dual mode OT is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{FakeSetup}, \mathsf{Extract}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$, *with the following syntax:*

- $\mathsf{Setup}(1^\lambda)$, *takes as input a security parameter, and outputs a* crs.
- $\mathsf{FakeSetup}(1^\lambda)$, *takes as input a security parameter, and outputs a* crs_S *and a trapdoor* td *that can be used to extract the sender's input.*
- $\mathsf{Extract}(td, ot_2)$, *takes as input the trapdoor* td, *and* any OT_2 *message* ot_2, *outputs the sender's input* $\{m_c\}_{c \in C}$.

– $\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3$ *have the same syntax as in Definition 1, except that they also take* crs *as input.*

The correctness, receiver security and statistical sender privacy properties are the same as Definition 1. We additionally require the following properties:

1. *CRS Indistinguishability*, we have

$$crs \overset{c}{\approx} crs_S,$$

where crs is generated by Setup, and crs_S is generated by $\mathsf{FakeSetup}$.

2. *Extraction Correctness*, for any receiver's input $b \in [C]$ and any unbounded adversary $\mathcal{A}$, we have

$$\Pr_{\substack{(crs_S, td) \leftarrow \mathsf{FakeSetup}(1^\lambda), \\ (ot_1, st) \leftarrow \mathsf{OT}_1(crs, b) \\ ot_2^* \leftarrow \mathcal{A}(crs, ot_1)}} \left[y \leftarrow \mathsf{OT}_3(crs, st, ot_2^*), \{m_c^*\}_{c \in [C]} \leftarrow \mathsf{Extract}(td, ot_2^*) : y = m_b^* \right] = 1.$$

Theorem 5 ([22]). *Assuming hardness of either LWE or DDH, there exists a black-box construction of dual-mode oblivious transfer.*

2.2 Dual-Mode Commitments

We recall the definition of a dual-mode public key encryption system [22]. Since in our application the default mode these crypto systems are instantiated in is the *lossy* mode, we refer to them by *dual-mode commitments.*

Definition 3. *A dual-mode commitment is a tuple of PPT algorithms* $\mathsf{Com} = (\mathsf{Gen}, \mathsf{FakeGen}, \mathsf{Commit}, \mathsf{Extract})$ *having the following interface*

– $\mathsf{Gen}(1^\lambda)$*, on input a security parameter* λ*, outputs a common reference string* crs*.*
– $\mathsf{FakeGen}(1^\lambda)$*, on input a security parameter* λ*, outputs a common reference string* crs *and an* extraction trapdoor td*.*
– $\mathsf{Commit}(crs, b)$*, on input a bit* $b \in \{0, 1\}$*, outputs a commitment* com*.*
– $\mathsf{Extract}(td, \tilde{t})$*, on input an extraction trapdoor* td*, and a commitment* com*, outputs a bit* $b \in \{0, 1\}$*.*

We require the scheme to satisfy the following properties

1. Extraction Correctness*, for any* $\lambda \in \mathbb{N}$ *and* $b \in \{0, 1\}$*,*

$$\Pr[\mathsf{Extract}(td, \tilde{t}) = b] = 1,$$

where, $(crs, td) \leftarrow \mathsf{Gen}(1^\lambda)$ *and* $\tilde{t} \leftarrow \mathsf{Commit}(crs, b)$*.*

2. Indistinguishable CRS Modes*, we have*

$$\{crs : crs \leftarrow \mathsf{Gen}(1^\lambda)\}_{\lambda \in \mathbb{N}} \overset{c}{\approx} \{crs : (crs, td) \leftarrow \mathsf{FakeGen}(1^\lambda)\}_{\lambda \in \mathbb{N}}$$

3. Statistical Hiding*, the following two distributions are statistically indistinguishable*

$$\{\mathsf{Commit}(crs, 0) : crs \leftarrow \mathsf{Gen}(1^\lambda)\}_{\lambda \in \mathbb{N}} \overset{s}{\approx} \{\mathsf{Commit}(crs, 1) : crs \leftarrow \mathsf{Gen}(1^\lambda)\}_{\lambda \in \mathbb{N}}$$

Theorem 6 ([22]). *Assuming hardness of either LWE or DDH, there exists a black-box construction of dual-mode commitments.*

2.3 Commit-and-Prove

We formulate the properties and the interface that we need from a commit-and-prove system. Then, we observe that the MPC-in-the-head paradigm can be used to build a commit-and-prove system with these properties.

Definition 4. *A commit-and-prove system with challenge space $\mathcal{C}$ for a language $L \in \mathsf{NP}$, is a tuple of algorithms $\Pi = (\mathsf{Setup}, \mathsf{FakeSetup}, \mathsf{Com}, \mathsf{GenFresh}, \mathsf{P1}, \mathsf{P2}, \mathsf{Verify}, \mathsf{Extract})$ having the following interface*

- $\mathsf{Setup}(1^\lambda)$, *on input a security parameter λ, outputs a common reference string crs.*
- $\mathsf{FakeSetup}(1^\lambda)$, *on input a security parameter λ, outputs a common reference string crs and an* extraction trapdoor *td.*
- $\mathsf{Com}(crs, w; r)$ *on input a bitstring $w \in \{0,1\}^W$ outputs a commitment $\tilde{w}$.*
- $\mathsf{GenFresh}(crs)$, *on input a common reference string crs, outputs a sequence of* fresh commitments *along with their corresponding randomness Γ.*
- $\mathsf{P1}(crs, x, \mathbf{w}, \mathbf{r}, \Gamma; r_P)$, *on input a common reference string crs, an instance $x \in \{0,1\}^\ell$, a witness $\mathbf{w} = \{w_i \in \{0,1\}^W\}_{i\in[c]}$, initial commitment randomness $\mathbf{r} = \{r_i\}_{i\in[c]}$, fresh commitments and their randomness Γ, and the random coins r_P, outputs the first part of proof string π_1.*
- $\mathsf{P2}(crs, x, \mathbf{w}, \mathbf{r}, \Gamma, r_P, ch)$, *on input the same parameters of* $\mathsf{P1}$*, the random coins used by* $\mathsf{P1}$*, and the challenge ch, outputs the second part of the proof string π_2.*
- $\mathsf{Verify}(crs, x, \{\tilde{w}_i\}_{i\in[c]}, ch, \pi_1, \pi_2)$, *on input a common reference string crs, an instance $x \in \{0,1\}^\ell$, a sequence of commitments $\{\tilde{w}_i\}_{i\in[c]}$, a challenge $ch \in \mathcal{C}$, and a proof string (π_1, π_2), either accepts or rejects.*
- $\mathsf{Extract}(td, \tilde{t})$, *on input an extraction trapdoor td, and a commitment $\tilde{t}$, outputs a plaintext $t \in \{0,1\}^W$.*

We further require the commit and proof system to satisfy the following properties.

- *Completeness*, for any instance $x \in L$, and any tuple of strings $(w_1, w_2, \ldots, w_c) \in \{0,1\}^{c\times W}$ which is a witness for x, let $\tilde{w}_i \leftarrow \mathsf{Com}(crs, w_i)$ be commitments to w_i, we have
$$\Pr_{\substack{crs\leftarrow\mathsf{Setup}(1^\lambda)\\ \mathsf{P1}(crs,x,\mathbf{w},\mathbf{r},\Gamma)\\ ch\leftarrow\mathcal{C}\\ \pi_2\leftarrow\mathsf{P2}(ch,\mathsf{st})}}\left[\mathsf{Verify}(crs, x, \{\tilde{w}_i\}_{i\in[c]}, ch, \pi_1, \pi_2) \text{ accepts}\right] = 1.$$
- *Indistinguishable CRS modes*, we have
$$crs \stackrel{c}{\approx} crs',$$
where crs is generated by the genuine setup $\mathsf{Setup}(1^\lambda)$, and crs' is generated by the fake setup $\mathsf{FakeSetup}(1^\lambda)$.

– *Statistical Hiding*, for any two sequences of bitstrings $w^0 = \{w^0\}_{\lambda\in\mathbb{N}}, w^1 = \{w^1\}_{\lambda\in\mathbb{N}}$, the commitments are statistically indistinguishable under the *genuine setup*, namely,

$$\{\mathsf{Com}(crs, w^0_\lambda) : crs \leftarrow \mathsf{Setup}(1^\lambda)\}_{\lambda\in\mathbb{N}} \overset{s}{\approx} \{\mathsf{Com}(crs, w^1_\lambda) : crs \leftarrow \mathsf{Setup}(1^\lambda)\}_{\lambda\in\mathbb{N}}.$$

– *ϵ-Soundness,* let $\mathcal{R}$ be the NP-relation for the language L. For any unbounded adversary $(\mathsf{P1}^*, \mathsf{P2}^*)$, after the following procedure,
  - Generate the fake CRS with trapdoor $(crs, td) \leftarrow \mathsf{FakeSetup}(1^\lambda)$
  - $(x, \{\tilde{w}_i\}_{i\in[c]}, \pi_1, \mathsf{st}) \leftarrow \mathsf{P1}^*(crs)$
  - Sample a random challenge $ch \leftarrow \mathcal{C}$
  - $\pi_2 \leftarrow \mathsf{P2}^*(ch, \mathsf{st})$

  we have

$$\Pr\left[\mathcal{R}(x, \{\mathsf{Extract}(td, \tilde{w}_i)\}_{i\in[c]}) \neq 1 \wedge \mathsf{Verify}(crs, x, \{\tilde{w}_i\}_{i\in[c]}, ch, \pi_1, \pi_2) \text{ accepts}\right] < \epsilon.$$

– *Special Statistical Zero-Knowledge,* there exists a simulator algorithm Sim, such that, under any crs sampled by the genuine Setup algorithm, for any family of instances $\{x_\lambda\}$ with $x_\lambda \in L$, any witness $\{w_{\lambda,i}\}_{i\in[c]}$ for x_λ, any challenge $ch \in \mathcal{C}$, we have

$$(\mathsf{Com}(crs, \{w_{\lambda,i}\}_{i\in[c]}; \mathbf{r}), \pi_1, \pi_2) \overset{s}{\approx} (c', \pi'_1, \pi'_2),$$

  where π_1, π_2 are the outputs of the honest prover's algorithm for the instance x_λ, witness $\{w_{\lambda,i}\}_{i\in[c]}$, initial commitment randomness $\mathbf{r}$, and challenge ch, and $(c', \pi'_1, \pi'_2) \leftarrow \mathsf{Sim}(x_\lambda, ch)$ is output by the simulator.

Theorem 7 (Black-Box Commit-and-Prove from MPC-in-the-Head). *There exists a commit-and-prove protocol with constant soundness error. Furthermore, the honest prover's algorithms* $(\mathsf{P1}, \mathsf{P2})$ *only use information-theoretic building-blocks. Moreover, if the NP-relation of L can be verified by a circuit of depth d, then the algorithms* $\mathsf{P1}, \mathsf{P2}$ *can also be computed by a circuit of depth $O(d)$.*

Proof (Proof Sketch). The work [13] constructed zero-knowledge from secure multiparty computation protocols. We use their zero-knowledge protocol to build a commit-and-prove system, and prove that it only makes black-box use of cryptography. We now describe the main algorithms.

– $\mathsf{Com}(crs, w; r)$: Let $n = O(1)$ be a constant. First, it secret shares the witness $w = w_1 \oplus w_2 \oplus \ldots w_n$ to n shares, and then commits to each share separately using a dual-mode commitment scheme.
– $\mathsf{P1}(crs, x, \mathbf{w}, \mathbf{r}, \Gamma; r_P)$: Let $R(\cdot,\cdot)$ be the relation circuit of the language L. It uses a semi-honest information theoretic multiparty computation scheme (MPC) in the dishonest majority setting [4] for n parties. For every $i \in [n]$, the ith party holds w_i as its input. The prover runs the MPC "in its head" to jointly compute $R(x, w_1 \oplus w_2 \oplus \ldots \oplus w_n) = 1$, and obtains the view of each party $\mathsf{View}_1, \mathsf{View}_2, \ldots, \mathsf{View}_n$. Then, it outputs commitments to the views.

- $ch \leftarrow \mathcal{C}$: The challenge ch represent two random parties $ch \leftarrow [n] \times [n]$.
- $\mathsf{P2}(crs, x, \mathbf{w}, \mathbf{r}, \Gamma, r_P, ch)$: The prover does the same computation as $\mathsf{P1}$, and then opens the commitment of the views specified by ch, and also opens the commitments to the shares specified by ch.
- Verify: The verifier checks
  - The openings of the commitments are correct.
  - The views are consistent. Namely, the messages sent and received have the same values.

The zero-knowledge and the soundness property follow from the security and the correctness of the underlying MPC scheme. Now, we show that the construction only makes black-box use of cryptography. Since the MPC is information theoretic, the only part that uses cryptography is the commitments in $\mathsf{P1}$. To make $\mathsf{P1}$ information theoretic, we provide it a series of fresh commitments to 0 and 1 and their randomness in Γ. Then we have the prover choose which commitment it needs to use. This makes $\mathsf{P1}$ information theoretic.

Now we analyze the depth of $\mathsf{P1}$. Let the depth of the circuit R be d. Since we only have a constant number of parties, the secret sharing of $\mathbf{w}$ needs a constant depth circuit. For each gate in R, we only need a constant depth circuit to compute the corresponding messages in the MPC. Hence, the computation of the views $\mathsf{View}_1, \mathsf{View}_2, \cdots, \mathsf{View}_n$ can be done in depth $O(d)$.

The depth of $\mathsf{P2}$ can also be bounded by $O(d)$. This is because it does the same computation as $\mathsf{P1}$, and an additional commitment opening in the end. The commitment opening is selecting the commitment randomness specified by ch. Hence, it can be computed by a constant depth circuit.

2.4 Maliciously Function Private Homomorphic Encryption

We review the definition of maliciously function private homomorphic encryption. Notice that in our abstraction of homomorphic encryption, secret keys are generated corresponding to fresh ciphertexts, and can only decrypt the evaluated versions of their corresponding fresh ciphertexts. The reason we choose this abstraction is that we want it to be consistent with the construction in [14]. We mention that this abstraction is sufficient for our use-case.

Definition 5 ([21]). *Let $\mathcal{F} = \{\mathcal{F}_{\lambda,L}\}_{\lambda,L\in\mathbb{N}}$ be a family of boolean functions, where for each $\lambda, L \in \mathbb{N}$, the functions in $\mathcal{F}_{\lambda,L}$ have input size $\ell(\lambda, L)$. A maliciously function private homomorphic encryption (HE) scheme for $\mathcal{F}$ is a tuple of algorithms*
$\mathsf{HE} = (\mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{Sim})$*, where, except for* Sim *the rest of the algorithms are PPT, having the following interfaces*

- $\mathsf{Enc}(1^\lambda, 1^L, m)$*, given a security parameter $\lambda \in \mathbb{N}$, a function family index $L \in \mathbb{N}$, and a message $m \in \{0,1\}^\ell$, outputs a ciphertext $ct \in \{0,1\}^{\ell_{ct}(\lambda,L)}$ and a private key sk.*
- $\mathsf{Eval}(ct, f)$*, given a ciphertext ct, and a boolean function $f : \{0,1\}^\ell \to \{0,1\}$, outputs an evaluated ciphertext $ct_{eval} \in \{0,1\}^{\ell_{eval}}$.*

– $\mathsf{Dec}(sk, ct)$, *given a secret key* sk *and a ciphertext* ct, *outputs a bit* $b \in \{0,1\}$.
– $\mathsf{Sim}(ct^*, b)$, *on input a ciphertext* $ct^* \in \{0,1\}^{\ell_{ct}(\lambda,L)}$, *and a bit* b, *outputs a simulated ciphertext* ct_{sim}.

We consider HE schemes that satisfy the following properties:

1. Completeness, *for every* $\lambda, L \in \mathbb{N}$, *every function* $f \in \mathcal{F}_{\lambda,L}$ *and every input* $m \in \{0,1\}^{\ell}$,
$$\Pr[\mathsf{Dec}(sk, ct_{eval}) = f(m)] = 1,$$
where, $(ct, sk) \leftarrow \mathsf{Enc}(1^{\lambda}, 1^{L}, m)$, *and* $ct_{eval} \leftarrow \mathsf{Eval}(ct, f)$.
2. Compactness, *there exists a fixed polynomial* $\ell_{eval} = \ell_{eval}(\lambda, L)$ *such that evaluated ciphertexts have size* $\ell_{eval}(\lambda, L)$, *i.e., the size of evaluated ciphertexts only depend on the index of the family of functions being evaluated.*
3. Semantic Security, *for every non-uniform polynomial-size adversary* $\mathcal{A}$, *every* $L \in \mathbb{N}$, *and every two sequence of message* $m^0 = \{m^0_\lambda \in \{0,1\}^{\ell(\lambda,L)}\}_{\lambda\in\mathbb{N}}$ *and* $m^1 = \{m^1_\lambda \in \{0,1\}^{\ell(\lambda,L)}\}_{\lambda\in\mathbb{N}}$ *the probabilities*
$$\Pr[\mathcal{A}(ct) = 1], \tag{1}$$
in the following two experiments differ by only $\mathrm{negl}(\lambda)$*:*
   – *in experiment 0,* $(ct, sk) \leftarrow \mathsf{Enc}(1^{\lambda}, 1^{L}, m^0_\lambda)$
   – *in experiment 1,* $(ct, sk) \leftarrow \mathsf{Enc}(1^{\lambda}, 1^{L}, m^1_\lambda)$
4. Malicious Function Privacy, *for every* $L \in \mathbb{N}$, *and every ciphertext* $ct^* \in \{0,1\}^{\ell_{ct}(\lambda,L)}$, *there exists a* $m^* \in \{0,1\}^{\ell(\lambda,L)}$ *such that, for every function* $f \in \mathcal{F}_{\lambda,L}$,
$$\mathsf{Eval}(ct^*, f) \overset{s}{\approx} \mathsf{Sim}(ct^*, f(m^*))$$
.

If we instantiate the rate-1 OT-based HE construction of [14] with the recent rate-1 statistical sender private OT of [1] we get a malicious function private HE for branching programs.

Theorem 8 ([14,1]). *Assuming either DDH or LWE, there exists a* black-box *construction of maliciously function private homomorphic encryption scheme for the function family* $\mathcal{B} = \{\mathcal{B}_L\}_{L\in\mathbb{N}}$, *where for each* $L \in \mathbb{N}$, $\mathcal{B}_L$ *is the set of branching programs of length* L.

If we slightly relax the black-box requirement, we can have a lattice-based leveled maliciously function private FHE scheme, i.e., a maliciously function private HE scheme supporting all bounded-depth polynomial circuits.

Theorem 9 ([10]). *Assuming LWE, there exists a leveled maliciously function private homomorphic encryption scheme. The non-black-box use of cryptography in this scheme is restricted to bootstrapping (which is needed for every evaluation).*

2.5 Somewhere Statistically Binding Hash

Here we define a variant of somewhere statistically binding hashes [12].

Definition 6. *Fix a word size $W = W(\lambda)$. A somewhere statistical binding hash scheme is a tuple of PPT algorithms* $\mathsf{SSB} = (\mathsf{Gen}, \mathsf{Hash}, \mathsf{Verify}, \mathsf{Extract})$ *with the following syntax.*

- $\mathsf{Gen}(1^\lambda, N, S)$, *on input a security parameter λ, a database size N, and a subset of indices $S \subseteq [N]$, outputs a hash key hk along with a trapdoor td.*
- $\mathsf{Hash}(hk, \mathsf{DB})$, *on input a hash key hk and a database* DB *of N words of size W, outputs a hash value h along with N openings $\{\tau_i\}_{i\in[N]}$.*
- $\mathsf{Verify}(hk, h, i, x, \tau)$, *on input a hash key hk, a hash value h, an index i, a word x, and an opening ρ, either accepts or rejects.*
- $\mathsf{Extract}(td, h)$, *on input a hash value h , and a trapdoor td, outputs entries $\{x_i\}_{i\in S}$.*

We require the scheme to satisfy the following properties:

1. Correctness, *for all $\lambda, N \in \mathbb{N}$, any subset of indices $S \subseteq [N]$, any index $i \in [N]$, and any database* DB *of size N, we have*
$$\Pr[\mathsf{Verify}(hk, h, i, \mathsf{DB}_i, \tau_i) \text{ accepts}] = 1,$$
where, $(hk, td) \leftarrow \mathsf{Gen}(1^\lambda, N, S)$ and $(h, \{\tau_i\}_{i\in[N]}) := \mathsf{Hash}(hk, \mathsf{DB})$.
2. Index Hiding, *for any two sets S_1, S_2 of the same size, we have*
$$crs_1 \overset{c}{\approx} crs_2,$$
where crs_1 is generated by $\mathsf{Gen}(1^\lambda, N, S_1)$, and crs_2 is generated by $\mathsf{Gen}(1^\lambda, N, S_2)$.
3. Extraction Correctness, *for all $\lambda, N \in \mathbb{N}$, any subset of indices $S \subseteq [N]$, any index $i \in [N]$, any database* DB *of size N, and any hash h, we have*
$$\Pr[\mathsf{Verify}(hk, h, i, \mathsf{DB}_i, \tau_i) \text{ accepts} \wedge x_i \neq \mathsf{DB}_i] = 0,$$
where, $(hk, td) \leftarrow \mathsf{Gen}(1^\lambda, N, S)$ and $\{x_i\}_{i\in[S]} := \mathsf{Extract}(td, h)$.
4. Efficiency: *any hash key* hk *and opening τ corresponding to size N databases and index sets of size $|S|$, are of size $|S| \cdot \log(N) \cdot \mathrm{poly}(\lambda)$. Further,* Verify *can be implemented by a circuit of size $|S| \cdot \log(N) \cdot \mathrm{poly}(\lambda)$.*

Our definition is slightly stronger than the one in [12] in that (i) our hashing key is binding to a subset of indices instead of binding to a single index and, (ii) we need perfect extractable binding instead of just statistical binding, i.e., there is a trapdoor that allows extracting the ith value for each binding index i. We can get the former property by repeating any single-index binding scheme multiple times in parallel. For the latter property, we notice that the HE-based construction in [12] already achieves this property, however, it is non-black-box due to the use of bootstrapping in the underlying HE. We observe that if we use a rate-1 OT scheme instead of HE, then, we have a black-box construction satisfying all the requirements in Definition 6.Please refer to the full version for a sketch of the construction.

Theorem 10. *Assuming hardness of either DDH or LWE, there exists a black-box construction of somewhere statistically binding hash satisfying the properties listed in Definition 6.*

3 Defining C-PSM

First, we formally define the relations we consider in our protocols.

Definition 7 (H-Instance Entropic Relations). *Let X and Y be two sets. Let $\mathcal{R} \subseteq X \times Y$ be a relation. For any distribution D on X, we say $\mathcal{R}$ is H-*instance entropic with respect to *D, if, for every $w \in Y$,*

$$\Pr_{x \leftarrow D}[(x, w) \in \mathcal{R}] \leq 2^{-H}.$$

Next, we define the search functionality.

Definition 8 (Search function). *Fix parameters $\ell, c, W, N \in \mathbb{N}$. The procedure Search takes as input a boolean function $f : \{0,1\}^\ell \times \{0,1\}^{c \cdot W} \to \{0,1\}$, a bitstring $x \in \{0,1\}^\ell$, and a database DB consisting of N words of size W. It either outputs the lexicographically first c indices $i_1, \cdots, i_c \in [N]$ such that $f(x, \mathsf{DB}_{i_1}, \cdots, \mathsf{DB}_{i_c}) = 1$ or $\perp$ if no such c indices exist.*

We are now ready to define C-PSM.

Definition 9 (2-Round C-PSM). *Let $\ell = \ell(\lambda), c = c(\lambda), W = W(\lambda)$ and $H = H(\lambda)$ be integer parameters. Let D be a distribution on $\{0,1\}^\ell$. Fix a family of H-instance entropic boolean functions $f = \{f_\lambda : \{0,1\}^{\ell(\lambda)} \times \{0,1\}^{c(\lambda) \cdot W(\lambda)} \to \{0,1\}\}$ with respect to D. A credible private set membership protocol for f, denoted by C-PSM, is a protocol between a sender and a receiver described by a tuple of PPT algorithms $(\mathsf{Setup}, \mathsf{R}, \mathsf{S}, \mathsf{Verify})$, with the following syntax:*

- $\mathsf{Setup}(1^\lambda, N)$*, on input a security parameter λ and database size N, outputs a CRS crs.*
- $\mathsf{R}(crs, x)$*, given a CRS crs and an input x, outputs a* receiver message *α and an internal state st.*
- $\mathsf{S}(crs, \alpha, \mathsf{DB})$*, on input a CRS crs, receiver message α, and database DB, outputs a sender message β.*
- $\mathsf{Verify}(\beta, st)$*, on input a sender message β and internal state st, either accepts or rejects.*

We require the protocol to satisfy the following properties

1. Correctness*, for every $\lambda, N \in \mathbb{N}$, every input $x \in \{0,1\}^\ell$, and every database DB of size N such that $\mathsf{Search}(f, x, \mathsf{DB}) \neq \perp$, we have*

$$\Pr_{\substack{crs \leftarrow \mathsf{Setup}(1^\lambda, N) \\ (\alpha, st) \leftarrow \mathsf{R}(crs, x) \\ \beta \leftarrow \mathsf{S}(crs, \alpha, \mathsf{DB})}}[\mathsf{Verify}(\beta, st) \text{ accepts}] = 1.$$

2. δ-Soundness, *for every non-uniform malicious sender* $\mathsf{S}^* = \{\mathsf{S}^*_\lambda\}_{\lambda\in\mathbb{N}}$, *and every* $\lambda, N \in \mathbb{N}$,

$$\Pr_{\substack{crs\leftarrow\mathsf{Setup}(1^\lambda,N)\\ x\leftarrow D\\ (\alpha,st)\leftarrow\mathsf{R}(crs,x)\\ \beta\leftarrow\mathsf{S}^*(crs,\alpha)}}[\mathsf{Verify}(\beta,st)\ \textit{accepts}] \le \delta(\lambda) + 2^{-H(\lambda)}$$

3. Receiver Privacy, *for any sequence of CRS strings* $crs = \{crs_\lambda\}_{\lambda\in\mathbb{N}}$, *and for any two sequence of input strings* $x^0 = \{x^0_\lambda\}_{\lambda\in\mathbb{N}}$, $x^1 = \{x^1_\lambda\}_{\lambda\in\mathbb{N}}$,

$$\{crs_\lambda, \alpha : (\alpha, st) \leftarrow \mathsf{R}(crs_\lambda, x^0_\lambda)\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \{crs_\lambda, \alpha : (\alpha, st) \leftarrow \mathsf{R}(crs_\lambda, x^1_\lambda)\}_{\lambda\in\mathbb{N}}.$$

4. Statistical Malicious Sender Privacy, *there is a (possibly unbounded) simulator algorithm* Sim, *such that, for every sequence of first message strings* $\alpha = \{\alpha_\lambda\}_{\lambda\in\mathbb{N}}$, *there exists a sequence of inputs* $x^* = \{x^*_\lambda\}$, *such that for any* $N \in \mathbb{N}$, *and for every database* DB *of* N *records, the following two distributions are statistically indistinguishable,*
   - *first, generate* $crs \leftarrow \mathsf{Setup}(1^\lambda, N)$, *output* $\mathsf{Sim}(crs_\lambda, \alpha_\lambda, x^*_\lambda, \mathsf{Search}(f, x^*_\lambda, \mathsf{DB}))$,
   - *first, generate* $crs \leftarrow \mathsf{Setup}(1^\lambda, N)$, *output* $\mathsf{S}(crs, \alpha_\lambda, \mathsf{DB})$.
5. Efficiency, *both* R *and* Verify *have runtime* $\mathrm{poly}(\lambda, \ell, c, W, \log(N))$.

Remark 1. Notice that the notion of sender privacy in in Definition 9 does not prevent leaking the indices for the witness in the database. This is W.L.O.G and merely for the ease of exposition. To prevent this leakage, the sender can simply randomly shuffle the entries in its database.

4 Construction for String Equality

Here we present the simplest version of our construction where the predicate is simply string equality, that is, the receiver wants to learn whether its input is in the sender's database. The resulting protocol has 2 rounds, achieves $\mathrm{negl}(\lambda)$-soundness in a single repetition, and, does not depend on a CRS. For this construction, let the input size and the database word size be equal, i.e., $\ell(\lambda) = W(\lambda) \ge \lambda$. Also, define D to be the uniform distribution on $\{0,1\}^\ell$. Observe that for strings of length ℓ, the string equality relation is an ℓ-instance entropic relation with respect to D.

We new describe the ingredients in our construction.

- The first ingredient is a one-way function $f : \{0,1\}^* \to \{0,1\}^*$. We assume f maps $\ell(\lambda)$-bit inputs to $m(\lambda)$-bit outputs.
- The second ingredient is a maliciously circuit private homomorphic encryption scheme $\mathsf{HE} = (\mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{Sim})$ for the class of branching programs $\mathcal{B} = \{\mathcal{B}_L\}_{\lambda,L\in\mathbb{N}}$. Where, for each $L \in N$, $\mathcal{B}_L$ consists of all branching programs of length L.

Construction 1. Let $L := L(\lambda, N)$ be the length of the branching program computing the function Find described in Figure 1. The construction is as follows:

– $\mathsf{R}(x)$:
  - Compute the image of x under f to obtain $y := f(x)$.
  - Encrypt y under HE to produce $(ct, sk) \leftarrow \mathsf{HE.Enc}(1^\lambda, 1^L, y)$.
  - Output $\alpha := ct$ and store internal state $st := sk$.

– $\mathsf{S}(\alpha, \mathsf{DB})$:
  - Parse $\alpha := ct$.
  - Apply f to every entry in DB to obtain $\widetilde{\mathsf{DB}} = \{\widetilde{\mathsf{DB}}_i := f(\mathsf{DB}_i)\}_{i \in [N]}$.
  - Homomorphically evaluate the function $\mathsf{Find}_{\widetilde{\mathsf{DB}},\mathsf{DB}}$ on ct to obtain $ct_{eval} \leftarrow \mathsf{HE.Eval}(ct, \mathsf{Find}_{\widetilde{\mathsf{DB}},\mathsf{DB}},)$.
  - Output $\beta := ct_{eval}$.

– $\mathsf{Verify}(\beta, st)$:
  - Parse β and st as $\beta = ct_{eval}$ and $st = sk$ respectively.
  - Decrypt ct_{eval} to obtain $\tilde{x} := \mathsf{HE.Dec}(sk, ct_{eval})$.
  - Accept iff $f(\tilde{x})$ equals $f(x)$.

procedure $\mathsf{Find}_{\widetilde{\mathsf{DB}},\mathsf{DB}}(y)$
  if $y \notin \widetilde{\mathsf{DB}}$ **then**
    Output $\perp$
  else
    Find the smallest index i such that $\widetilde{\mathsf{DB}}_i = y$.
    Output DB_i.

Fig. 1: Description of the labeled-PSM functionality Find

Correctness and receiver privacy of Construction 1 immediately follows from the correctness and semantic security of HE. For efficiency, we have to argue that the length of the branching program computing Find is logarithmic in N. To do this, as shown in [5], we can convert the database DB to a trie, and essentially implement Find by a branching program of length ℓ.

We now prove the soundness of Construction 1.

Theorem 11. *Assuming f is one-way, Construction 1 is* $\text{negl}(\lambda)$*-sound.*

Proof. Let S^* be a malicious sender. Denote the success probability of S^* by p. In more detail, p is defined as

$$p := \Pr_{\substack{x \leftarrow D \\ (ct,sk) \leftarrow \mathsf{HE.Enc}(1^\lambda, 1^L, f(x)) \\ \beta \leftarrow S^*(ct) \\ \tilde{x} := \mathsf{HE.Dec}(sk,\beta)}} [f(\tilde{x}) = f(x)].$$

We use S^* to build a PPT adversary $\mathcal{A}$ which breaks the one-wayness of f with probability p. $\mathcal{A}$ works as follows, on input an image y, it first encrypts y by HE to obtain $(ct, sk) \leftarrow \mathsf{HE.Enc}(1^\lambda, 1^L, y)$. It then runs S^* on input ct to get $ct_{eval} \leftarrow S^*(ct)$. Finally, $\mathcal{A}$ decrypts ct_{eval} using sk and outputs $\tilde{x} := \mathsf{HE.Dec}(sk, ct_{eval})$ as the preimage of y. Now observe that as long as y is an image of an input chosen from the distribution D, the view of S^* when interacting with $\mathcal{A}$ is identical to its view in the soundness game. Therefore,

$$\Pr_{\substack{x \leftarrow D \\ y := f(x) \\ \tilde{x} \leftarrow \mathcal{A}(y)}} [f(\tilde{x}) = f(x)] = p.$$

This completes the proof.

Theorem 12. *Assuming* HE *is maliciously circuit private, Construction 1 satisfies statistical malicious sender privacy.*

Proof. Let α be an arbitrary first message and DB be any database of size $N \in \mathbb{N}$. We only describe the simulator algorithm Sim, the theorem follows instantly from the malicious function privacy of HE.

- Sim receives as input a first message $\alpha := ct$, and a bitstring x^*.
- Using the HE simulator it computes $ct_{eval} \leftarrow \mathsf{HE.Sim}(ct, x^*)$.
- It outputs ct_{eval}.

5 Construction for Predicates Beyond String Equality

Now we consider richer families of predicates. Fix input length $\ell = \ell(\lambda)$, word size $W = W(\lambda)$, function arity $c = c(\lambda)$, distribution D on $\{0,1\}^\ell$, and entropy parameter $H = H(\lambda)$. Let $f : \{0,1\}^\ell \times \{0,1\}^{c \cdot W} \to \{0,1\}$ be an H-instance entropic function with respect to D.

In the rest of the paper, we construct a 2-round C-PSM protocol in three steps.

- First, we construct a 4-round protocol satisfying a weaker notion of soundness, where, it is only required that an adversary cannot convince a verifier for any *fixed* set of indices.
- Then, using dual-mode 2-round OT, we show how to compress the 4-round protocol to a 2-round protocol which still has weak soundness.
- Finally, we amplify the soundness of the 2-round protocol by parallel repetition to achieve a (strongly) sound 2-round protocol.

5.1 Weakly-Sound 4-Round Protocol

We first construct a *weakly-sound* 4-round protocol with constant soundness. Where a weakly-sound 4-round C-PSM protocol is defined as follows:

Definition 10 (Weakly-Sound 4-Round C-PSM). *A credible private set membership protocol with challenge space $\mathcal{C}$ for f is a protocol between a sender and a receiver described by a tuple of PPT algorithms* (Setup, R, S1, S2, Verify)*, with the following syntax:*

- $\mathsf{Setup}(1^\lambda, N)$*, on input a security parameter λ and database size N, outputs a CRS crs.*

- $\mathsf{R}(crs, x)$, *given a CRS crs and an input x, outputs a* receiver message α *and an internal state* st_R.
- $\mathsf{S1}(crs, \alpha, \mathsf{DB})$, *on input a CRS crs, a receiver message α, and a database* DB, *outputs a sender message β_1 and an internal state st_S.*
- $\mathsf{S2}(crs, ch, st_S)$, *on input a CRS crs, a challenge ch, and an internal state st_S, outputs a sender message β_2.*
- $\mathsf{Verify}(\beta_1, ch, \beta_2, st_R)$ *on input sender messages β_1, β_2, challenge ch, and internal state st_R, either accepts and outputs a sequence $S = \{i_k\}_{k \in [c]}$ of indices, or, rejects.*

We require the protocol to satisfy the following properties

1. Correctness, *for every $\lambda, N \in \mathbb{N}$, every input $x \in \{0,1\}^\ell$, every database* DB *of size N such that* $\mathsf{Search}(f, x, \mathsf{DB}) \neq \bot$, *and every challenge $ch \in \mathcal{C}$, we have*
$$\Pr_{\substack{crs \leftarrow \mathsf{Setup}(1^\lambda, N) \\ (\alpha, st_R) \leftarrow \mathsf{R}(crs, x) \\ (\beta_1, st_S) \leftarrow \mathsf{S1}(crs, \alpha, \mathsf{DB}) \\ \beta_2 \leftarrow \mathsf{S2}(crs, ch, st_S)}} [\mathsf{Verify}(\beta_1, ch, \beta_2, st_R) \text{ accepts}] = 1.$$
2. Weak δ-Soundness, *for every non-uniform malicious sender* $\mathsf{S}^* = \{(\mathsf{S1}^*_\lambda, \mathsf{S2}^*_\lambda)\}_{\lambda \in \mathbb{N}}$, *every $\lambda, N \in \mathbb{N}$, and every sequence of indices $I^* = \{i^*_k\}_{k \in [c]}$ of size c,*
$$\Pr_{\substack{crs \leftarrow \mathsf{Setup}(1^\lambda, N) \\ x \leftarrow D \\ (\alpha, st_R) \leftarrow \mathsf{R}(crs, x) \\ (\beta_1, st_S) \leftarrow \mathsf{S1}^*(crs, \alpha) \\ ch \leftarrow \mathcal{C} \\ \beta_2 \leftarrow \mathsf{S2}^*(crs, ch, st_S)}} [\mathsf{Verify}(\beta_1, ch, \beta_2, st_R) = I^*] \leq \delta(\lambda) + 2^{-H(\lambda)}$$
3. Receiver Privacy, *for any sequence of CRS strings $crs = \{crs_\lambda\}_{\lambda \in \mathbb{N}}$, and for any two sequence of input strings $x^0 = \{x^0_\lambda\}_{\lambda \in \mathbb{N}}$, $x^1 = \{x^1_\lambda\}_{\lambda \in \mathbb{N}}$,*
$$\{crs_\lambda, \alpha : (\alpha, st) \leftarrow \mathsf{R}(crs_\lambda, x^0_\lambda)\}_{\lambda \in \mathbb{N}} \overset{c}{\approx} \{crs_\lambda, \alpha : (\alpha, st) \leftarrow \mathsf{R}(crs_\lambda, x^1_\lambda)\}_{\lambda \in \mathbb{N}}.$$
4. Special Statistical Malicious Sender Privacy, *there is a simulator algorithm* Sim, *such that, for every sequence of first message strings $\alpha = \{\alpha_\lambda\}_{\lambda \in \mathbb{N}}$, there exists a sequence of inputs $x^* = \{x^*_\lambda\}$, such that for every database* DB, *and for every $ch \in \mathcal{C}$, the following two distributions are statistically indistinguishable*
   - *sample* $crs \leftarrow \mathsf{Setup}(1^\lambda, N)$, *then, output* $\mathsf{Sim}(crs, x^*_\lambda, ch, \mathsf{Search}(f, x^*_\lambda, \mathsf{DB}))$
   - *sample* $crs \leftarrow \mathsf{Setup}(1^\lambda, N)$, *then, generate* $(\beta_1, st) \leftarrow \mathsf{S1}(crs, \alpha_\lambda)$, *next, compute* $\beta_2 \leftarrow \mathsf{S2}(crs, ch, st)$, *finally, output* (β_1, β_2).
5. Efficiency, *both* R *and* Verify *have runtime* $\mathrm{poly}(\lambda, \ell, c, W, \log(N))$.

Our construction uses the following ingredients:

- A commit-and-prove system $\Pi = (\mathsf{Setup}, \mathsf{FakeSetup}, \mathsf{Com}, \mathsf{GenFresh}, \mathsf{P}, \mathsf{Verify}, \mathsf{Extract})$ for the language specified by f.

– A maliciously circuit private homomorphic encryption scheme $\mathsf{HE} = (\mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{Sim})$ for a class of functions $\mathcal{F} = \{\mathcal{F}_L\}_{L\in\mathbb{N}}$.
– A somewhere statistically binding hash $\mathsf{SSB} = (\mathsf{Gen}, \mathsf{Hash}, \mathsf{Verify}, \mathsf{Extract})$ satisfying the properties in Definition 6.

Construction 2 (Weakly-Sound 4-Round C-PSM). Let $L := L(\lambda, N)$ be a function family index such that $\mathcal{F}_L$ includes both G^1 and G^2 for databases DB of size N. The construction is as follows:

– $\mathsf{Setup}(1^\lambda, N)$:
  - Generate a CRS for Π, $crs_\Pi \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$.
  - Generate an SSB hash key binding to the first c indices (or any other arbitrary sequence of c indices), $(hk, td) \leftarrow \mathsf{SSB.Gen}(1^\lambda, N, \{i\}_{i\in[c]})$.
  - Output $crs := (crs_\Pi, hk)$.
– $\mathsf{R}(crs, x)$:
  - Encrypt x under HE to produce $(ct, sk) \leftarrow \mathsf{HE.Enc}(1^\lambda, 1^L, x)$.
  - Output $\alpha := ct$ and store internal state $st := sk$.
– $\mathsf{S1}(crs, \alpha, \mathsf{DB})$:
  - Parse crs and α as (crs_Π, hk) and ct respectively.
  - Commit to every entry in DB to produce $\widetilde{\mathsf{DB}} = \{\widetilde{\mathsf{DB}}_i \leftarrow \Pi.\mathsf{Com}(crs_\Pi, \mathsf{DB}_i; r_i^{com})\}_{i\in[N]}$.
  - Hash $\widetilde{\mathsf{DB}}$ using SSB to obtain $(h, \{\tau_i\}_{i\in[N]}) := \mathsf{SSB.Hash}(hk, \widetilde{\mathsf{DB}})$.
  - Produce fresh commitments and their randomness $\Gamma \leftarrow \Pi.\mathsf{GenFresh}(crs_\Pi)$.
  - Sample random coins r_P for $\Pi.\mathsf{P1}$.
  - Homomorphically evaluate the function G^1 on ct to obtain $ct_{eval,1} \leftarrow \mathsf{HE.Eval}(crs_\Pi, ct, G^1_{\mathsf{DB},\widetilde{\mathsf{DB}},\{\tau_i\}_{i\in[N]},\{r_i^{com}\}_{i\in[N]},\Gamma,r_P})$.
  - Output $\beta_1 := (h, ct_{eval,1})$ and store internal state $st := (x, \mathsf{DB}, \{r_i^{com}\}_{i\in[N]}, \Gamma, r_P)$.
– $\mathsf{S2}(crs, ch, st)$:
  - Parse crs and st as (crs_Π, hk) and $(x, \mathsf{DB}, \{r_i^{com}\}_{i\in[N]}, \Gamma, r_P)$ respectively.
  - Homomorphically evaluate the function G^2 on ct to obtain $ct_{eval,2} \leftarrow \mathsf{HE.Eval}(crs_\Pi, ct, G^2_{crs_\Pi,\mathsf{DB},\{r_i^{com}\}_{i\in[N]},\Gamma,r_P,ch})$.
  - Output $\beta_2 := ct_{eval,2}$.
– $\mathsf{Verify}(crs, \beta_1, ch, \beta_2, st)$:
  - Parse crs,β_1,β_2 and st as (crs_Π, hk), $(h, ct_{eval,1})$, $ct_{eval,2}$ and sk respectively.
  - Decrypt $ct_{eval,1}$ to obtain $(\{i_k\}_{k\in[c]}, \{\tilde{w}_k\}_{k\in[c]}, \pi_1, \{\tau_k\}_{k\in[c]}) := \mathsf{HE.Dec}(sk, ct_{eval,1})$.
  - Decrypt $ct_{eval,2}$ to obtain $\pi_2 := \mathsf{HE.Dec}(sk, ct_{eval,2})$.
  - Accept and output $\{i_k\}_{k\in[c]}$ iff $\Pi.\mathsf{Verify}(crs_\Pi, x, \{\tilde{w}_k\}_{k\in[c]}, ch, \pi_1, \pi_2)$ accepts and
    $\forall k \in [c] : \mathsf{SSB.Verify}(hk, h, i_k, \tilde{w}_k, \tau_k)$ accepts.

We first prove δ-soundness and special statistical malicious sender privacy of Construction 2.

Theorem 13. *Assuming* SSB *is index-hiding,* Π *has indistinguishable CRS modes,* HE *has semantic security, and* Π *is* δ*-sound, Construction 2 is weakly* $(\delta + \gamma)$*-sound for any positive constant (or any non-negligible function)* γ.

procedure $G^1_{crs_\Pi,\mathsf{DB},\widetilde{\mathsf{DB}},\{\tau_i\}_{i\in[N]},\Gamma,r_P}(x)$
Let $out := \mathsf{Search}(x, f, \mathsf{DB})$
if $out == \bot$ **then**
Output $\bot$
else
Parse out as $out = (i_1, \cdots, i_c)$.
Generate the first prover message:

$$\pi_1 \leftarrow \mathsf{P1}(crs_\Pi, x, \{\mathsf{DB}_{i_k}\}_{k\in[c]}, \{r_{i_k}\}_{k\in[c]}, \Gamma; r_P).$$

Output $(\{i_k\}_{k\in[c]}, \{\widetilde{\mathsf{DB}}_{i_k}\}_{k\in[c]}, \pi_1, \{\tau_{i_k}\}_{k\in[c]})$.

Fig. 2: Description of G^1

procedure $G^2_{crs_\Pi,\mathsf{DB},\{r_i^{com}\}_{i\in[N]},\Gamma,r_P,ch}(x)$
Let $out := \mathsf{Search}(x, f, \mathsf{DB})$
if $out == \bot$ **then**
Output $\bot$
else
Parse out as $out = (i_1, \cdots, i_c)$.
Generate the second prover message:

$$\pi_2 \leftarrow \mathsf{P2}(crs_\Pi, x, \{\mathsf{DB}_{i_k}\}_{k\in[c]}, \{r_{i_k}\}_{k\in[c]}, \Gamma, r_P, ch).$$

Output the second prover message π_2.

Fig. 3: Description of G^2

Proof. Let $\mathsf{S}^* = (\mathsf{S1}^*, \mathsf{S2}^*)$ be a malicious sender and let $I^* = \{i_k^*\}_{k\in[c]}$ be any sequence of indices of size c. For each hybrid H_j, define the probability p_j as follows:

$$p_j := \Pr[\Pi.\mathsf{Verify}(crs_\Pi, x, \{\tilde{w}_k\}_{k\in[c]}, \pi_1, ch, \pi_2) \text{ accepts } \wedge \forall k \in [c] : \mathsf{SSB.Verify}(hk, h, i_k^*, \tilde{w}_k, \tau_k) \text{ accepts}].$$

where in each hybrid we describe how crs_Π, x, $\{\tilde{w}_k\}_{k\in[c]}$, π_1, ch, π_2, hk, h, and $\{\tau_k\}_{k\in[c]}$ are defined.

Hybrid H_0**:** This is the soundness experiment. In more detail, here,

- $crs_\Pi \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$,
- $(hk, td_{SSB}) \leftarrow \mathsf{SSB.Gen}(1^\lambda, N, \{k\}_{k\in[c]})$,
- $x \leftarrow D$,
- $(ct, sk) \leftarrow \mathsf{HE.Enc}(1^\lambda, 1^L, x)$,
- $((h, ct_{eval,1}), st) \leftarrow \mathsf{S1}^*((crs_\pi, hk), ct)$,
- $ch \leftarrow \mathcal{C}$,
- $ct_{eval,2} \leftarrow \mathsf{S2}^*(crs, ch, st)$,
- $(\{i_k\}_{k\in[c]}, \{\tilde{w}_k\}_{k\in[c]}, \pi_1, \{\tau_k\}_{k\in[c]}) := \mathsf{HE.Dec}(sk, ct_{1,eval})$,
- and $\pi_2 := \mathsf{HE.Dec}(sk, ct_{eval,2})$.

Hybrid H_1: This is identical to H_0 except that here hk is generated binding to indices $i_1^*, \cdots, i_c^*$, i.e., $(hk, td_{ssb}) \leftarrow \mathsf{SSB.Gen}(1^\lambda, N, \{i_k^*\}_{k\in[c]})$. The index hiding property of SSB implies that $H_0 \overset{c}{\approx} H_1$. Consequently, $|p_0 - p_1| = \text{negl}(\lambda)$.

Hybrid H_2: The only difference between this hybrid and H_1 is that here, crs_Π is generated along with a trapdoor td_Π via $(crs_\Pi, td_\Pi) \leftarrow \Pi.\mathsf{FakeSetup}(1^\lambda)$. Since Π has indistinguishable CRS modes, $H_1 \overset{c}{\approx} H_2$. Therefore, $|p_1 - p_2| = \text{negl}(\lambda)$.

Lemma 1. *Assuming* HE *is semantically secure,* $p_2 - (\delta + 2^{-H}) = \text{negl}(\lambda)$.

Proof. Using S^* we build an adversary $\mathcal{A}$ against the semantic security of HE. $\mathcal{A}$ works as follows:

- It generates crs_Π, hk, and td_{ssb} exactly as in H_2.
- It samples two elements $x_0 \leftarrow D, x_1 \leftarrow D$.
- $\mathcal{A}$ sends x_0, x_1 to the semantic security challenger of HE.
- It receives as response an HE ciphertext ct from the HE semantic security challenger. The ciphertext ct either encrypts x_0 or x_1 under an honestly generated HE key sk.
- $\mathcal{A}$ runs $\mathsf{S1}^*$ to obtain $((h, ct_{eval,1}), st) \leftarrow \mathsf{S1}^*((crs_\Pi, hk), ct)$
- $\mathcal{A}$ receives a random challenge $ch \leftarrow \mathcal{C}$.
- $\mathcal{A}$ runs $\mathsf{S2}^*$ to obtain $ct_{eval,2} \leftarrow \mathsf{S2}^*((crs_\Pi, hk), st)$.
- Using td_{ssb} it recovers commitments $\{\tilde{w}_k^*\}_{k\in[c]} := \mathsf{SSB.Extract}(td_{ssb}, h)$. Using td_{com}, for each $k \in [c]$ it recovers $w_k^* := \mathsf{Com.Extract}(td_{com}, \tilde{w}_k^*)$.
- If $f(x_0, \{w_k^*\}_{k\in[c]}) = 1$, it outputs 1. Otherwise, it outputs 0.

Now we analyze the success probability of $\mathcal{A}$ in breaking the semantic security of HE. Let

$$(\{i_k\}_{k\in[c]}, \{\tilde{w}_k\}_{k\in[c]}, \pi_1, \{\tau_k\}_{k\in[c]}) := \mathsf{HE.Dec}(sk, ct_{eval,1}).$$

First, we consider the case where ct encrypts x_0. In this case with probability at least p_2,

$$\forall k \in [c] : \mathsf{SSB.Verify}(hk, h, i_k^*, \tilde{w}_k, \tau_k) \text{ accepts,} \tag{2}$$

and

$$\Pi.\mathsf{Verify}(crs_\Pi, x_0, \{\tilde{w}_k\}_{k\in[c]}, \pi_1, ch, \pi_2) \text{ accepts.} \tag{3}$$

By extractability of SSB, the former implies that $\forall k \in [c] : \tilde{w}_k = \tilde{w}_k^*$. Consequently, by δ-soundness of Π, with probability at least $p_2 - \delta$, $f(x_0, \{w_k^*\}_{k\in[c]}) = 1$. We conclude that in this case $\mathcal{A}$ outputs 1 with probability at least $p_2 - \delta$. Now we turn to the other case where ct encrypts x_1. In this case, x_0 maintains all of its entropy , therefore, since f is H-instance entropic,

$$\Pr[f(x_0, \{w_k^*\}_{k\in[c]}) = 1] = 2^{-H},$$

i.e., $\mathcal{A}$ outputs 1 with probability 2^{-H}. We showed that $\mathcal{A}$ breaks the semantic security of HE with probability at least $p_2 - \delta - 2^{-H}$.

This concludes the proof.

Theorem 14. *Assuming* HE *is maliciously circuit private, Π satisfies special statistical zero-knowledge, and Π has statistically hiding commitments, Construction 2 satisfies special statistical malicious sender privacy.*

Proof. Let α be an arbitrary first message, $ch \in \mathcal{C}$ be any challenge, DB be any database of size $N \in \mathbb{N}$, and let $crs \leftarrow \mathsf{Setup}(1^\lambda, N)$ be a crs generated through Setup. First, we describe the simulator algorithm Sim.

- Sim receives as input a CRS parsed as $crs := (crs_\Pi, hk)$, a first message $\alpha := ct$, a bitstring x^*, and indices $\{i_k^*\}_{k\in[c]}$ (W.L.O.G assume that the indices are not $\perp$).
- Using the zero-knowledge simulator for Π, it computes $(\{\tilde{w}_k^*\}_{k\in[c]}, \pi_1^*, \pi_2^*) \leftarrow \Pi.\mathsf{Sim}(crs_\Pi, x, ch)$.
- For each $i \in [N]/\{i_k^*\}_{k\in[c]}$, Sim computes a commitment $\widetilde{\mathsf{DB}}_i \leftarrow \Pi.\mathsf{Commit}(crs_{com}, \mathbf{0})$. For each $k \in [c]$ it sets the i_k^*th commitment to be equal to $\widetilde{\mathsf{DB}}_{i_{k^*}} := \tilde{w}_k^*$.
- It hashes $\widetilde{\mathsf{DB}}$ to obtain $(h, \{\tau_i\}_{i\in[N]}) := \mathsf{SSB.Hash}(hk, \widetilde{\mathsf{DB}})$.
- Using the HE simulator it computes

$$ct_{eval,1} \leftarrow \mathsf{HE.Sim}(ct, (\{i_k^*\}_{k\in[c]}, \{\tilde{w}_k^*\}_{k\in[c]}, \pi_1^*, \{\tau_{i_k^*}\}_{k\in[c]})).$$

- Using the HE simulator it computes

$$ct_{eval,2} \leftarrow \mathsf{HE.Sim}(ct, \pi_2^*)$$

- It outputs $(h, ct_{eval,1}, ct_{eval,2})$.

We now proceed via a series of hybrids to show that the output of Sim is statistically indistinguishable from an honestly generated sender message.

Hybrid H_0: This hybrid corresponds to generating the sender messages β_1, β_2 honestly through $(\beta_1, st) := (h, ct_{eval}) \leftarrow \mathsf{S1}(crs, \alpha, \mathsf{DB})$ and $\beta_2 := ct_{eval} \leftarrow \mathsf{S2}(crs, ch, st)$.

Hybrid H_1: This hybrid uses HE.Sim to produce $ct_{eval,1}$ and $ct_{eval,2}$. In more detail, given ct, we know that there exists an x^* such that,

$$\mathsf{HE.Eval}(ct, G^1_{crs_\Pi, \mathsf{DB}, \widetilde{\mathsf{DB}}, \{\tau_i\}_{i\in[N]}, \Gamma, r_P}) \overset{s}{\approx} \mathsf{HE.Sim}(ct, G^1_{crs_\Pi, \mathsf{DB}, \widetilde{\mathsf{DB}}, \{\tau_i\}_{i\in[N]}, \Gamma, r_P}(x^*)),$$

and

$$\mathsf{HE.Eval}(ct, G^2_{crs_\Pi, \mathsf{DB}, \{r_i^{com}\}_{i\in[N]}, \Gamma, r_P, ch}) \overset{s}{\approx} \mathsf{HE.Sim}(ct, G^2_{crs_\Pi, \mathsf{DB}, \{r_i^{com}\}_{i\in[N]}, \Gamma, r_P, ch}(x^*)).$$

In this hybrid, $ct_{eval,1}$ and $ct_{eval,2}$ are generated as

$$ct_{eval,1} \leftarrow \mathsf{HE.Sim}(ct, G^1_{crs_\Pi, \mathsf{DB}, \widetilde{\mathsf{DB}}, \{\tau_i\}_{i\in[N]}, \Gamma, r_P}(x^*)),$$

and

$$ct_{eval,2} \leftarrow \mathsf{HE.Sim}(ct, G^2_{crs_\Pi,\mathsf{DB},\{r_i^{com}\}_{i\in[N]},\Gamma,r_P,ch}(x^*)).$$

It follows from the malicious circuit privacy of HE that $H_0 \overset{s}{\approx} H_1$.

Hybrid H_2: The difference between this hybrid and the previous hybrid is only syntactical. In this hybrid, to generate $ct_{eval,1}$ and $ct_{eval,2}$, first, the (lexicographically) smallest indices $\{i_k^*\}_{k\in[c]}$ such that $f(x^*, \{\mathsf{DB}_{i_k^*}\}_{k\in[c]}) = 1$ are computed. Next, π_1 and π_2 are computed as

$$\pi_1 \leftarrow P1(crs_\Pi, x, \{\mathsf{DB}_{i_k^*}\}_{k\in[c]}, \{r_{i_k^*}\}_{k\in[c]}, \Gamma; r_P)$$

and

$$\pi_2 \leftarrow P2(crs_\Pi, x, \{\mathsf{DB}_{i_k^*}\}_{k\in[c]}, \{r_{i_k}\}_{k\in[c]}, \Gamma, r_P, ch).$$

Finally, $ct_{eval,1}$ and $ct_{eval,2}$ are computed as

$$ct_{eval,1} \leftarrow \mathsf{HE.Sim}(ct, (\{i_k^*\}_{k\in[c]}, \{\widetilde{\mathsf{DB}}_{i_k^*}\}_{k\in[c]}, \pi_1, \{\tau_{i_k^*}\}_{k\in[c]})),$$

and

$$ct_{eval,2} \leftarrow \mathsf{HE.Sim}(ct, \pi_2).$$

As already stated H_1 and H_2 are identical.

Hybrid H_3: In this hybrid we modify how $\widetilde{\mathsf{DB}}$ is generated. Here, for each $k \in [c]$,

$$\widetilde{\mathsf{DB}}_{i_k^*} \leftarrow \Pi.\mathsf{Commit}(crs_\Pi, \mathsf{DB}_{i_k^*}; r_{i_k^*}^{com})$$

as before, but the rest of the commitments are generated as

$$\{\widetilde{\mathsf{DB}}_i \leftarrow \Pi.\mathsf{Commit}(crs_\Pi, \mathbf{0})\}_{i\in[N]/\{i_k^*\}_{k\in[c]}}.$$

Notice that we don't modify the commitments whose randomness are used in the HE.Sim algorithm. Therefore, by the statistical hiding property of the commitments in Π, $H_2 \overset{s}{\approx} H_3$.

Hybrid H_4: The difference between this hybrid and the previous hybrid is that here $\{\widetilde{\mathsf{DB}}_{i_k^*}\}_{k\in[c]}, \pi_1$, and π_2 are generated using the simulator for Π, i.e.,

$$(\{\widetilde{\mathsf{DB}}_{i_k^*}\}_{k\in[c]}, \pi_1, \pi_2) \leftarrow \Pi.\mathsf{Sim}(x^*, ch).$$

The special zero-knowledge property of Π directly implies that $H_3 \overset{s}{\approx} H_4$. Observe that, H_4 corresponds to generating the sender messages via Sim.

Depending on how HE is instantiated, Construction 2 can support different classes of predicates with different trade-offs in terms of black-box usage of underlying cryptographic primitives. If we instantiate HE with Theorem 8, we can have a black-box construction supporting NC^1 predicates f where $\mathsf{Search}(\cdot, f, \mathsf{DB})$ can be implemented in by a branching program whose length is logarithmic in $|\mathsf{DB}|$.

Theorem 15. *Assuming hardness of either of DDH or LWE, there exists a family of weakly-sound 4-round C-PSM protocols with the following properties:*

1. *It supports all predicates f such that f can be implemented by an NC^1 circuit and also for every database* DB *of size N,* $\mathsf{Search}(\cdot, f, \mathsf{DB})$ *can be implemented by a branching program of length logarithmic in N.*
2. *It only makes black-box use of the underlying cryptographic primitives.*
3. *It is receiver private.*
4. *It is weakly δ-sound.*
5. *It satisfies special statistical malicious sender privacy.*

Proof. We instantiate Construction 2 with the black-box maliciously circuit private homomorphic encryption scheme of Theorem 8 for the class of branching programs $\{\mathcal{B}_L\}_{L\in\mathbb{N}}$. We have already proven weak δ-soundness and special statistical malicious sender privacy of Construction 2. Correctness follows from the correctness of HE, correctness of Π, and correctness of SSB. Receiver privacy follows from the semantic security of HE. For efficiency, we need to show that both G^1 and G^2 can be evaluated by a branching program of length $L = \text{poly}(\lambda, \log N)$. Observe that both G^1 and G^2 access the whole database only through the Search functionality. Therefore, since the Search functionality for f can be implemented by a branching program of length logarithmic in N, L is also logarithmic in N. Furthermore, since f is in NC^1, by Theorem 7, both P1 and P2 are also in NC^1. Consequently, by Barrington's theorem [3], P1 and P2 can be implemented by a polynomial (in λ) length branching program. Therefore, $L = \text{poly}(\lambda, \log N)$.

Alternatively, we can instantiate HE with Theorem 9 to get a construction supporting all bounded depth circuits. While this construction only makes black-box use of HE, however, the homomorphic encryption scheme constructed in Theorem 9 is non-black-box due to relying on bootstrapping.

Theorem 16. *Assuming hardness of LWE, there exists a family of weakly-sound 4-round C-PSM protocols with the following properties:*

1. *It supports all predicates f such that f can be implemented by bounded-depth circuits, i.e., the C-PSM protocol is leveled.*
2. *Its only non-black-box use of the underlying cryptographic primitives happens through bootstrapping.*
3. *It is receiver private.*
4. *It is weakly δ-sound.*
5. *It satisfies special statistical malicious sender privacy.*

Proof. We instantiate Construction 2 with the maliciously circuit private homomorphic encryption scheme of Theorem 9 for the class of circuits $\{\mathcal{F}_L\}_{L\in\mathbb{N}}$, where for each $L \in \mathbb{N}$, $\mathcal{F}_L$ consists of all circuits of depth at most L. Establishing weak δ-soundness, special statistical malicious sender privacy, correctness and receiver privacy is identical to Theorem 15. For efficiency, it is straightforward to verify that G^1 and G^2 can be evaluated by circuits of depth $L = \text{poly}(\lambda, \log N)$.

5.2 4-Round to 2-Round Transformation

Here we provide a generic transformation that converts any weakly-sound 4-round C-PSM protocol to a weakly-sound 2-round protocol. Analogously to weakly-sound 4-round C-PSM, we define weakly-sound 2-round C-PSM as follows:

Definition 11 (Weakly Sound 2-Round C-PSM). *Let ℓ, c, W, f, H, D be the same as Definition 9. A weakly sound C-PSM for f, is a protocol between a sender and a receiver described by a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{R}, \mathsf{S}, \mathsf{Verify})$*, where the interface of* $\mathsf{Setup}, \mathsf{R}$ *and* S *is identical to their interface in Definition 9 and* Verify *has the following syntax:*

– $\mathsf{Verify}(\beta, st)$*, on input a sender message β and internal state st, either accepts and outputs a sequence $I = \{i_k\}_{k\in[c]}$ of indices, or rejects.*

Except for δ-soundness we require the protocol to satisfy all properties in Definition 9. Additionally, we consider the following weaker variant of soundness:

1. Weak δ-Soundness*, for every non-uniform malicious sender $\mathsf{S}^* = \{\mathsf{S}^*_\lambda\}_{\lambda\in\mathbb{N}}$, every $\lambda, N \in \mathbb{N}$, and every sequence of indices $I^* = \{i^*_k\}_{k\in[c]}$ of size c,*

$$\Pr_{\substack{crs\leftarrow\mathsf{Setup}(1^\lambda,N)\\ x\leftarrow D\\ (\alpha,st)\leftarrow\mathsf{R}(crs,x)\\ \beta\leftarrow\mathsf{S}^*(crs,\alpha)}}[\mathsf{Verify}(\beta, st) = I^*] \leq \delta(\lambda) + 2^{-H(\lambda)}$$

Our transformation uses the following ingredients:

– A 4-round weakly sound C-PSM protocol $\Sigma = (\mathsf{Setup}, \mathsf{R}, \mathsf{S1}, \mathsf{S2}, \mathsf{Verify})$.
– A dual-mode statistically sender private OT scheme $\mathsf{OT} = (\mathsf{Setup}, \mathsf{FakeSetup}, \mathsf{Extract}, \mathsf{OT1}, \mathsf{OT2}, \mathsf{OT3})$.

Construction 3. The construction is as follows:

– $\mathsf{Setup}(1^\lambda, N)$:
  - Generate a CRS for Σ, $crs_\Sigma \leftarrow \Sigma.\mathsf{Setup}(1^\lambda, N)$.
  - Generate a CRS for dual-mode OT, $crs_{OT} \leftarrow \mathsf{OT}.\mathsf{Setup}(1^\lambda)$.
  - Output $crs := (crs_\Sigma, crs_{OT})$.
– $\mathsf{R}(crs, x)$:
  - Generate a Σ first message for x along with an internal state, $(\alpha_\Sigma, st_\Sigma) \leftarrow \Sigma.\mathsf{R}(crs_\Sigma, x)$.
  - Sample a random challenge $ch \leftarrow \mathcal{C}$ from the challenge space of Σ.
  - Generate an OT first message for ch along with an internal state, $(ot_1, st_{OT}) \leftarrow \mathsf{OT}.\mathsf{OT1}(crs_{OT}, ch)$.
  - Output first message $\alpha := (\alpha_\Sigma, ot_1)$ and internal state $st = (x, ch, st_\Sigma, st_{OT})$.
– $\mathsf{S}(crs, \alpha, \mathsf{DB})$:
  - Parse crs and α as (crs_Σ, crs_{OT}) and (α_Σ, ot_1) respectively.
  - Compute $(\beta_1, st) \leftarrow \Sigma.S1(crs_\Sigma, \alpha_\Sigma, \mathsf{DB})$.
  - For each $ch \in \mathcal{C}$ compute $\beta_{2,ch} \leftarrow \Sigma.S2(crs_\Sigma, st, ch, \mathsf{DB})$.

- Compute OT second message $ot_2 \leftarrow \mathsf{OT.OT2}(ot_1, \{\beta_{2,ch}\}_{ch \in \mathcal{C}})$.
- Output $\beta := (\beta_1, ot_2)$.

– $\mathsf{Verify}(\beta, st)$:
  - Parse β and st as (β_1, ot_2) and $(x, ch, st_\Sigma, st_{OT})$ respectively.
  - Recover $\beta_{2,ch}$ as $\beta_{2,ch} := \mathsf{OT.OT3}(ot_2, st_{OT})$.
  - Output whatever $\Sigma.\mathsf{Verify}(\beta_1, ch, \beta_{2,ch}, st_\Sigma)$ outputs.

The correctness immediately follows from the correctness of Σ and OT. If the size of the challenge space of $\mathcal{C}$ is a constant (or scales logarithmically with N) then, the efficiency also directly follows from the efficiency of Σ. In the full version of this paper we prove the following two theorems.

Theorem 17 (Weak δ-Soundness). *Assuming Σ satisfies weak δ-soundness, Construction 3 satisfies weak $(\delta + \gamma)$-soundness for every constant (or even non-negligible function) $\gamma > 0$.*

Theorem 18 (Statistical Malicious Sender Privacy). *Assuming* OT *is statistical sender private, and Σ satisfies special statistical malicious sender privacy, Construction 3 is statistically malicious circuit private.*

5.3 Weakly δ-Sound to negl(λ)-Sound Transformation

Here we present a generic transformation that for any constant $\delta > 0$ converts a weakly δ-sound 2-round C-PSM to a negl(λ)-sound 2-round C-PSM. The transformation is essentially parallel repetition of the weakly-sound protocol, but the verification algorithm also checks that all the repetitions return the same set of indices.

For the following construction, let $\Sigma = (\mathsf{Setup}, \mathsf{R}, \mathsf{S}, \mathsf{Verify})$ be any weakly sound 2-round C-PSM with δ-soundness.

Construction 4. Let $rep := rep(\lambda, N, c)$ be a parameter indicating the number of repetitions.The construction is as follows:

– $\mathsf{Setup}(1^\lambda, N)$:
  - Generate and output rep independent CRSs for Σ, $crs := \{crs^i_\Sigma \leftarrow \Sigma.\mathsf{Setup}(1^\lambda, N)\}_{i \in [rep]}$.

– $\mathsf{R}(crs, x)$:
  - Generate rep first messages for Σ along with their internal state, $\{(\alpha^i_\Sigma, st^i_\Sigma) \leftarrow \Sigma.\mathsf{R}(crs^i_\Sigma, x)\}_{i \in [rep]}$.
  - Output the first messages $\alpha := \{\alpha^i_\Sigma\}_{i \in [rep]}$ and internal state $st = (x, \{st^i_\Sigma\}_{i \in [rep]})$.

– $\mathsf{S}(crs, \alpha, \mathsf{DB})$:
  - Compute and output rep second messages for Σ, $\beta := \{\beta^i_\Sigma \leftarrow \Sigma.\mathsf{S}(crs^i_\Sigma, \alpha^i_\Sigma, \mathsf{DB})\}_{i \in [rep]}$.

– $\mathsf{Verify}(\beta, st)$:
  - Accept iff each repetition accepts and outputs a sequence of indices of size c $\{I_i := \Sigma.\mathsf{Verify}\beta_i, st_i\}_{i \in [rep]}$ and all the sequences I_i are equal.

The correctness and statistical malicious sender privacy of Construction 4 immediately follow because the same properties hold in Σ. This construction satisfies efficiency as long as rep grows at most logarithmically in N.

Theorem 19. *If Σ is weakly δ-sound, then, Construction 4 is $N^c \cdot \delta^{rep}$-sound.*

Proof. For each possible sequence I^*, the probability that all of the repetitions accept and output I^* is at most δ^{rep}. Since we have at most N^c different sequences, the theorem follows.

By setting $rep := (\lambda + c \cdot \log(N))/\log(1/\delta)$ we get $2^{-\lambda}$ soundness.

5.4 Putting Everything Together

In this section, we combine the constructions in subsection 5.1 with the transformations in subsection 5.2 and subsection 5.3, to obtain 2-round C-PSM constructions for richer classes of functionalities.

Theorem 20. *Assuming hardness of either of DDH or LWE, there exists a family of 2-round C-PSM protocols in the CRS model with the following properties:*

1. *It supports all predicates f such that f can be implemented by an NC^1 circuit and also for every database* DB *of size N,* $\mathsf{Search}(\cdot, f, \mathsf{DB})$ *can be implemented by a branching program of length logarithmic in N.*
2. *It only makes black-box use of the underlying cryptographic primitives.*
3. *It is receiver private.*
4. *It is (strongly) sound.*
5. *It satisfies statistical malicious sender privacy.*
6. *It has transparent setup, i.e., the CRS is simply a random string.*

Theorem 21. *Assuming hardness of LWE, there exists a family of 2-round C-PSM protocols in the CRS model with the following properties:*

1. *It supports all predicates f such that f can be implemented by bounded-depth circuits, i.e., the C-PSM protocol is leveled.*
2. *Its only non-black-box use of the underlying cryptographic primitives happens through bootstrapping.*
3. *It is receiver private.*
4. *It is (strongly) sound.*
5. *It satisfies statistical malicious sender privacy.*
6. *It has transparent setup.*

References

1. Aggarwal, D., Döttling, N., Dujmovic, J., Hajiabadi, M., Malavolta, G., Obremski, M.: Algebraic restriction codes and their applications. In: ITC. pp. 2:1–2:15 (2022)
2. Apple Inc: Password monitoring - apple support (2021), https://support.apple.com/guide/security/password-monitoring-sec78e79fc3b/web

3. Barrington, D.A.M.: Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In: STOC. pp. 1–5 (1986)
4. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC. pp. 1–10 (1988)
5. Chase, M., Garg, S., Hajiabadi, M., Li, J., Miao, P.: Amortizing rate-1 OT and applications to PIR and PSI. TCC (2021)
6. Chen, H., Huang, Z., Laine, K., Rindal, P.: Labeled PSI from fully homomorphic encryption with malicious security. In: CCS. pp. 1223–1237 (2018)
7. Chen, H., Laine, K., Rindal, P.: Fast private set intersection from homomorphic encryption. In: CCS. pp. 1243–1255 (2017)
8. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. J. ACM **45**(6), 965–981 (1998)
9. Cong, K., Moreno, R.C., da Gama, M.B., Dai, W., Iliashenko, I., Laine, K., Rosenberg, M.: Labeled PSI from homomorphic encryption with reduced computation and communication. In: CCS. pp. 1135–1150 (2021)
10. Döttling, N., Dujmovic, J.: Maliciously circuit-private FHE from information-theoretic principles. In: ITC (2022)
11. Google Inc: Protect your accounts from data breaches with password checkup (2019), https://security.googleblog.com/2019/02/protect-your-accounts-from-data.html
12. Hubácek, P., Wichs, D.: On the communication complexity of secure function evaluation with long output. In: ITCS. pp. 163–172 (2015)
13. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: STOC. pp. 21–30 (2007)
14. Ishai, Y., Paskin, A.: Evaluating branching programs on encrypted data. In: TCC. pp. 575–594 (2007)
15. Izabachène, M., Nitulescu, A., de Perthuis, P., Pointcheval, D.: Myope: Malicious security for oblivious polynomial evaluation. In: SCN. pp. 663–686 (2022)
16. Kannepalli, S., Laine, K., Moreno, R.C.: Password monitor: Safeguarding passwords in microsoft edge (2021), https://www.microsoft.com/en-us/research/blog/password-monitor-safeguarding-passwords-in-microsoft-edge/
17. Marlinspike, M.: The difficulty of private contact discovery (2014), https://whispersystems.org/blog/contact-discovery/
18. Merkle, R.C.: A digital signature based on a conventional encryption function. In: CRYPTO. pp. 369–378 (1987)
19. Micali, S., Rabin, M.O., Kilian, J.: Zero-knowledge sets. In: FOCS. pp. 80–91 (2003)
20. Okamoto, T., Pietrzak, K., Waters, B., Wichs, D.: New realizations of somewhere statistically binding hashing and positional accumulators. In: ASIACRYPT. pp. 121–145 (2015)
21. Ostrovsky, R., Paskin-Cherniavsky, A., Paskin-Cherniavsky, B.: Maliciously circuit-private FHE. In: CRYPTO. pp. 536–553 (2014)
22. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: CRYPTO. pp. 554–571 (2008)