Info-Commit: Information-Theoretic Polynomial Commitment and Verification

Saeid Sahraei[1] and Salman Avestimehr[2]

[1] Qualcomm Technologies, Inc., San Diego, CA 92121, USA,
[2] University of Southern California, Los Angeles, CA 90089, USA.

Abstract. We introduce Info-Commit, a protocol for polynomial commitment and verification. Info-Commit consists of two phases. An initial commitment phase and an evaluation phase. During the commitment phase, the verifier and the prover engage in a private two-party computation algorithm so that the verifier extracts a private verification key. In the evaluation phase, the verifier is interested in learning the evaluations of the polynomial at several input points. Info-Commit has four main features. Firstly, the verifier is able to detect, with high probability, if the prover has responded with evaluations of the same polynomial that he has initially committed to. Secondly, Info-Commit provides rigorous privacy guarantees for the prover: upon observing the initial commitment and the response provided by the prover to m evaluation requests, the verifier only learns $O(m^2)$ symbols about the coefficients of the polynomial. Thirdly, the verifiability guarantee is unconditional and without the need for a trusted party, while "bounded storage" is the only assumption underlying the privacy of the algorithm. In particular, both properties hold regardless of the computation power of the two parties. Lastly, Info-Commit is doubly-efficient in the sense that in the evaluation phase, the verifier runs in $O(\sqrt{d})$ and the prover runs in $O(d)$, where $d-1$ is the degree of the polynomial.

1 Introduction

Consider a server hosting a closed-source program that is intended to solve a specific problem (say, an advanced optimization tool). As a customer, you wish to delegate the task of running this program on your data to the server. Naturally, your limited access to the software as well as your lack of trust in the server concerns you about the validity of the returned results. How can you verify that the server is indeed running the correct program on your data and providing you with the desired outcome? To convince yourself that the server is in possession of *some* program that solves the claimed problem you could *probe* the server at random input values for which you know the outcome (say, through a different commercial software to which you only have trial access). Nevertheless, the bigger concern is whether the server continues to run the correct program on your data once this initial trust has been established. From an economic perspective, the server may have the incentive not to spend his computational resources on your

data, but rather provide you with random outputs. Worse yet, a malicious server may intentionally provide you with false evaluations if he knows that you do not check the correctness of the results.

The literature on verifiable computing [1–4] and functional commitment protocols [5, 6] allow the server (prover) to generate a proof of correctness of his results which can be checked by the users (verifiers). However, verifying such proofs is only possible if one has substantial knowledge about the underlying function, or if a verification key has been generated by a trusted party. Furthermore, the soundness of these algorithms relies on heuristic hardness assumptions that may be falsified at any time due to algorithmic breakthroughs or technological advancements.

In this paper, our goal is to design a commitment and verification algorithm that overcomes these challenges. Namely, (i) the verifiability property must hold regardless of the computation power of the prover, (ii) there must be no need for a trusted party, and (iii) the verifier must not learn more than a constant number of bits about the underlying function, regardless of his computation power. In the example above, the prover should be able to keep his program private from the verifier while at the same time provide him with verifiable results. This verification is performed against an initial commitment by the prover which contains almost no information about the program itself. Once the verifier receives the commitment to a specific function, the prover will only be able to pass the verification process if he provides evaluations of the same function that he has committed to. Our focus will be specifically on functions that can be represented as polynomials. This encompasses a wide range of applications including transaction verification in cryptocurrencies [7], verifiable secret sharing [8, 6], and proof-of-storage [9, 10, 4].

1.1 Our contributions

We propose Info-Commit, a protocol for polynomial commitment and verification. Specifically, Info-Commit consists of a commitment phase and and evaluation phase. During the commitment phase, the prover and the verifier engage in a private two-party computation algorithm in such a way that the verifier learns a private commitment to the polynomial. In the evaluation phase, the verifier is interested in learning the evaluations of the polynomial at many input points with the help of the prover. Info-Commit is verifiable, doubly-efficient and privacy-preserving. More specifically, Info-Commit has the following salient features.

1. **Verifiability.** If the prover commits to a polynomial f in the commitment phase, he must provide evaluations of the same polynomial f in the evaluation phase. Otherwise, the verifier is able to detect the inconsistency with overwhelming probability. On the other hand, if the prover is honest, the verifier accepts the results with probability 1.
2. **Double-efficiency.** For each round of evaluation, the verifier runs in $O(\sqrt{d})$ whereas the prover runs in $O(d)$. Note that we measure the efficiency of Info-Commit in the amortized model [2] where the initial commitment phase is

followed by many rounds of evaluations. As a result, the one-time cost of the commitment phase can be neglected.

3. **Privacy.** The other facet of Info-Commit is its privacy-preserving property. Upon observing the initial commitment and the response of the prover to m rounds of evaluation, the verifier only learns $O(m^2)$ symbols over the field about the coefficients of the polynomial.
4. **Information-theoretic guarantees without relying on any trusted party.** Both the verifiability and the privacy of Info-Commit are information-theoretic, meaning that the verifier and the prover can have unbounded computation power and may arbitrarily deviate from the prescribed protocol. Furthermore, Info-Commit does not rely on any trusted party, not even in the initial commitment phase.

The only assumption that underlies the privacy of Info-Commit is the bounded storage assumption which states that there exists an upper-bound N on the storage size of the verifier. This upper bound must be publicly known and must not exceed n^2, where n is the minimum required storage for each party for the protocol to succeed. This assumption is used in the commitment phase where we rely on oblivious transfer algorithms in the bounded storage model [11–14].

1.2 Related work

The closest works in the literature to the present paper are commitment protocols, including polynomial commitment [6, 15–17], vector commitment [18], cryptographic accumulators [19], and more generally functional commitments [5]. These protocols allow a prover to provide a cryptographic commitment to a specific function which reveals little to no information about the underlying function. A verifier can then request the prover to "open" the commitment at specific locations. The verifier should be able to detect if the opening is inconsistent with the initial commitment. There are several factors that distinguish the model in the current paper from the traditional notion of functional commitment.

The *first* difference is in the notion of privacy. The linear commitment model in [5] requires the initial commitment to reveal no information about the function. However, it does not impose any requirement on how much information is revealed upon observing the evaluations (opening) of the functions along with the initial commitment. On the other hand, the polynomial commitment scheme in [6] only requires the *evaluations* of the polynomial at unqueried points to remain hidden from the verifier, which is a rather weaker privacy constraint. In contrast, we impose rigorous *information-theoretic* constraints on how much information is revealed about the *coefficients* of the polynomial upon observing the initial commitment *and* the evaluations at a constant number of input points.

Secondly, our model does not permit the assumption of a trusted party, a component that is common in functional commitment protocols [6, 5]. This trusted party is usually in charge of generating a public key (that is later used for commitment, evaluation and verification) in a *setup* phase which is eliminated in our model.

Thirdly, while the literature on functional commitment is more concerned with the size of the commitment and the size of the opening, we are instead focused on designing doubly-efficient algorithms, i.e., algorithms with efficient provers and super-efficient verifiers. Furthermore, unlike the literature on functional commitment, our verifiability is information-theoretic.

Another relevant concept is zero-knowledge verifiable computation [20] and zero-knowledge arguments of knowledge [21–24]. Such algorithms enable a verifier and a prover to interact in order to compute $f(v, u)$ where f is a publicly known function, v is the input of the verifier and u is the private input of the prover. The verifier will not learn anything about u except for what is implied through $f(v, u)$. Furthermore, the verifier will be convinced that there exists some u known to the prover such that the computed value corresponds to $f(v, u)$. For instance, f could correspond to a Rank-1 Constraint System (R1CS) [23], v a public input, and u a private witness such that (v, u) satisfies all the quadratic constraints of the R1CS. In the context of polynomial evaluation, v could be the input to the polynomial, u the coefficients of the polynomial, and f an operator that maps (v, u) to the evaluation of the polynomial. Unfortunately, this approach does not provide any binding guarantees, as the prover may change his polynomial for every input.

Other related notions in the literature are oblivious polynomial evaluation [25–27] which guarantees the privacy of both parties in a semi-honest setting, and verifiable computation [2, 4, 28] which guarantees efficient verifiability but generally ignores the privacy of the prover.

2 Polynomial Commitment and Verification

Suppose a prover is in possession of a private polynomial $f(x) = a_0 + a_1 x + \cdots + a_{d-1} x^{d-1}$ selected uniformly at random among all polynomials of degree $d-1$ over $\mathbb{F}_q$. A verifier, who only knows the degree of the polynomial, wishes to delegate the evaluation of this polynomial at several input points $x \in \mathbb{F}_q$ to the prover and verify the correctness of the results in sublinear time in d. Before this evaluation phase, the prover commits to the polynomial f during an initialization phase. Once this commitment is done, the prover is expected to evaluate the same polynomial f for every input x that is provided by the verifier. The verifier should be able to detect, efficiently and with high probability, if for any input $x \in \mathbb{F}_q$ the prover instead returns $y \neq f(x)$. We assume that the verifier is interested in evaluating the polynomial f at many input points, so that the complexity of the commitment phase amortizes over many rounds of the evaluation phase [2]. Therefore, efficiency is only measured with respect to the evaluation phase. Additionally, during the entire commitment and evaluation phase, the verifier should learn close to nothing about the coefficients of the polynomial f. Note that revealing a constant number of symbols over $\mathbb{F}_q$ about the coefficients of f to the verifier is inevitable, since such amounts of information can be learned by investigating a single pair $(x, f(x))$. Formally, a polynomial commitment and verification protocol consists of the following algorithms.

– **Commit**$(a_0, \cdots, a_{d-1}, K_v, K_p)$. In the commitment phase the verifier and the prover engage in a secure two-party computation algorithm commit$(a_0, \cdots, a_{d-1}, K_v, K_p)$ to help the verifier learn a secret verification key VK which depends on his secret key, K_v, the prover's secret key, K_p, and the polynomial coefficients $a_0, \cdots, a_{d-1}$.
– **Eval**$(x, a_0, \cdots, a_{d-1}, K_p)$. The verifier reveals a desired input x to the prover. The prover will then provide $val =$**Eval**$(x, a_0, \cdots, a_{d-1}, K_p)$ to help the verifier evaluate $f(x)$.
– **Verify**(x, val, VK, K_v). The verifier checks the correctness of val based on his secret key, K_v, and the output of the commitment phase, VK. If val is incompatible with the initial commitment, he will reject the evaluation. Otherwise he will proceed to recovering $f(x)$.
– **Recovery**(x, val). If the verification process passes, the verifier will proceed to recover the evaluation $f(x)$ via an algorithm **Recovery**(x, val).

The only assumption that underlies the privacy of Info-Commit is the bounded storage assumption stated as follows.

Definition 1 (The bounded storage assumption [13, 14]). *This assumption states that there exists an upper-bound N on the storage size of the verifier. Furthermore, the storage size of both the verifier and the prover is at least $N^{\frac{1}{2}+\epsilon}$. The upper-bound N is publicly known.*

We are interested in polynomial commitment and verification protocols that are private (in the bounded storage model) and verifiable as defined below.

Definition 2. *A polynomial commitment and verification protocol is information-theoretically private and verifiable if it satisfies the following properties.*

– **Correctness**. *If the prover follows the* **Commit** *and* **Eval** *algorithms for the same polynomial f, then the* **Verify** *algorithm must return 1 with probability 1.*
– **(Information-theoretic) Soundness**. *If the prover commits to a polynomial with coefficients $a_0, \cdots, a_{d-1}$, then the probability that he can pass the verification test with $\hat{val} \neq$* **Eval**$(x, a_0, \cdots, a_{d-1}, K_p)$ *should be negligible, regardless of his computation power.*

$$\mathbb{P}(\textbf{Verify}(x, \hat{val}, VK, K_v, K_p) = 1, \hat{val} \neq \textbf{Eval}(x, a_0, \cdots, a_{d-1}, K_p)) \approx 0. \tag{1}$$

– **Efficient verification**. *The two functions* **Verify** *and* **Recovery** *must run in sublinear time in d.*
– **Efficient evaluation**. *The running time of* **Eval** *must be comparable to the time required for evaluating the polynomial f, i.e., linear in d.*
– **(Information-theoretic) Privacy**. *After running* **Commit** *and* **Eval** *for m different inputs $x_1, \cdots, x_m$, the verifier should only learn $\rho = O(poly(m))$ symbols over the field about the coefficients of the polynomial f, regardless of his computation power but under the bounded storage assumption. Importantly, ρ should be independent of the degree of the polynomial.*

$$H_q(a_0, \cdots, a_{d-1} | \, VK, (x_1, val_1), \cdots, (x_m, val_m)) = d - O(poly(m)). \tag{2}$$

3 Preliminaries

In the analysis of security and privacy of Info-Commit, we make use of several basic properties of the entropy of random variables which we review in this section. Furthermore, in the commitment phase of Info-Commit, we rely on a secure two-party computation algorithm. Here we explain how this can be accomplished via a series of reductions to a much simpler problem known as one-out-of-two Oblivious Transfer [29, 30]. An information-theoretic solution for the latter will be outlined in the bounded storage model borrowed from [31, 11, 14, 13].

3.1 Entropy of random variables

In this section we overview the definitions of various entropy functions of random variables, the notion of Markov chain, and some useful properties that will help us establish the soundness and verifiability of Info-Commit.

Definition 3 (Entropy). *Let X, Y, Z be three random variables with domains $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ respectively. Let q be an arbitrary positive number.*

- *The entropy of X in base q is a measure of its randomness and is defined as*

$$H_q(X) = -\sum_{x \in \mathcal{X}} \mathbb{P}(X = x) \log_q \mathbb{P}(X = x). \tag{3}$$

  Note that for a uniformly distributed X over $\mathcal{X}$ we have $H_q(X) = log_q|\mathcal{X}|$ whereas for a deterministic X we have $H_q(X) = 0$.
- *The joint entropy of X and Y is defined as*

$$H_q(X, Y) = -\sum_{(x,y) \in \mathcal{X} \times \mathcal{Y}} \mathbb{P}(X = x, Y = y) \log_q \mathbb{P}(X = x, Y = y). \tag{4}$$

- *The conditional entropy of X given Y is defined as*

$$H_q(X|Y) = H_q(X, Y) - H_q(Y). \tag{5}$$

Some useful properties of the entropy terms are as follows.

- Property 1. All the entropy terms defined above are non-negative.
- Property 2. $\max\{H_q(X), H_q(Y)\} \leq H_q(X, Y) \leq H_q(X) + H_q(Y)$. This implies that conditioning cannot increase the entropy, $H(X|Y) \leq H(X)$.
- Property 3. If X is independent of (Y, Z) then $H_q(Y|X, Z) = H_q(Y|Z)$.

Definition 4 (Markov Chain). *An ordered set of three random variables (X, Y, Z) with domain $\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ is said to form a Markov chain if*

$$\mathbb{P}(X = x, Z = z|Y = y) = \mathbb{P}(X = x|Y = y)\mathbb{P}(Z = z|Y = y) \tag{6}$$

for every $(x, y, z) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$. Intuitively, this means that the two random variables X and Y are independent, conditioned on knowing Y. This Markov chain is represented via $X \longleftrightarrow Y \longleftrightarrow Z$.

- Property 4. If $X \longleftrightarrow Y \longleftrightarrow Z$ forms a Markov chain, then $H_q(X|Y, Z) = H_q(X|Y)$.

3.2 Information-theoretic Oblivious Transfer

Suppose Alice has c secrets $a_0, \cdots, a_{c-1}$ and Bob has a secret index $i \in [0 : c-1]$. Bob is interested in learning a_i without revealing any information about i to Alice. On the other hand, Alice wishes to reveal no information about $a_0, \cdots, a_{c-1}$ to Bob except for the value of a_i. This problem is referred to as Oblivious Transfer (OT) [30]. If each of Alice's c secrets belongs to an alphabet $\mathcal{Y}$ of size $|\mathcal{Y}| = q$, we refer to the problem as $\binom{c}{1}_q$−OT. In this paper we make use of an $\binom{c}{1}_q$−OT algorithm in the commitment phase. Information-theoretic algorithms for $\binom{2}{1}_q$ and $\binom{2}{1}_2$ have been proposed in the literature in the bounded storage model [13, 11, 14]. Furthermore, there exist unconditional reductions from $\binom{c}{1}_q$−OT to $\binom{2}{1}_q$−OT and from $\binom{c}{1}_q$−OT to $\binom{c}{1}_2$−OT [32]. We will provide brief overviews of these algorithms here and refer the readers to the corresponding references for a more detailed description.

Reduction from $\binom{c}{1}_q$−OT to $\binom{2}{1}_q$−OT. Information-theoretic reductions from $\binom{c}{1}_q$ to $\binom{c}{1}_2$ and from $\binom{c}{1}_2$ to $\binom{2}{1}_2$ have been presented in [32]. Here we describe a reduction from $\binom{c}{1}_q$−OT to $\binom{2}{1}_q$−OT and refer the interested reader to [32] to see the other reductions. Alice generates $c-2$ random symbols over the alphabet $\mathcal{Y}$, namely $r_0, \cdots, r_{c-3}$. Alice and Bob agree on an ordering of the elements of $\mathcal{Y}$. Let y_i be the i'th element of $\mathcal{Y}$ in this ordering for $i \in [0 : q-1]$. Define the summation operation over $\mathcal{Y}$ as $y_i + y_j = y_{mod(i+j,q)}$. Bob relies on $\binom{2}{1}_q$−OT to request one element of each column of Table 1. If Bob is interested

Index	Value 0	Value 1	$\cdots$	Value $c-3$	Value $c-2$
1	a_0	$a_1 + r_0$	$\cdots$	$a_{c-3} + r_{c-4}$	$a_{c-2} + r_{c-3}$
2	r_0	$r_0 + r_1$	$\cdots$	$r_{c-4} + r_{c-3}$	$a_{c-1} + r_{c-3}$

Table 1. Unconditional reduction from $\binom{c}{1}_q$−OT to $c-1$ invocations of $\binom{2}{1}_q$−OT.

in a_i, he will choose the second row for every column $j < i$, but chooses the first row for the i'th column. Bob's request to the remaining columns does not make any difference in the overall outcome: Bob will only learn a_i and nothing else about the remaining a_j values. If Bob wants a_{c-1}, he will choose the second row for every column of the table. Privacy of Bob follows from the privacy of the underlying $\binom{2}{1}_q$−OT algorithm.

$\binom{2}{1}_q$−OT in the bounded storage model. Several works have proposed $\binom{2}{1}_q$−OT and $\binom{2}{1}_2$−OT algorithms in the bounded storage model which remain secure even if the two parties have infinite computation power [14, 12, 13, 11]. It is only necessary to assume that an upper bound N on the storage size of

Bob is known to Alice. The actual required storage size n for the algorithm to succeed is much smaller than N. For instance, the algorithm in [14] only needs $n = N^{1/2+\epsilon}$. The general idea behind all these algorithms is as follows. Alice generates a random string of K bits such that $K > \alpha N$ for some $\alpha > 1$. Let R_X represent the sub-sequence of the random string indexed by the set $X \subseteq [N]$. Alice broadcasts this sequence, and each of Alice and Bob randomly stores a subset of the sequence of size $\sqrt{2\ell N}$. Let $\Omega_A \subseteq [N]$ represent the indices of the bits stored by Alice and define Ω_B similarly. By the birthday paradox, with high probability $|\Omega_A \cap \Omega_B| \approx \ell$. Alice reveals Ω_A to Bob and Bob computes two sets $\Omega_A \cap \Omega_B$ and $\Omega_B \backslash \Omega_A$. Next, the two parties engage in an interactive hashing algorithm [13] to generate two sets $X_0, X_1 \subseteq \Omega_A$ of equal size such that $X_i \subseteq \Omega_B$ but $|X_{1-i} \cap \Omega_B| \ll |X_{1-i}|$ for some $i \in \{0, 1\}$ chosen by Bob. The interactive hashing algorithm also guarantees that Alice does not learn i. Alice encodes a_0 with R_{X_0} and a_1 with R_{X_1} and sends both to Bob. Bob will be able to decode a_i, since he knows R_{X_i}. However, his incomplete knowledge of the set $R_{X_{1-i}}$ prevents him from learning a_{1-i}.

While the state of the art $\binom{2}{1}_q-$OT and $\binom{2}{1}_2-$OT algorithms [14, 13] guarantee perfect privacy for Bob, they only guarantee Alice's privacy with probability $1 - 2^{-k^c}$ for some security parameter k and an arbitrary constant c. The negligible probability that Alice's privacy is compromised will translate to a negligible probability that Info-Commit fails to guarantee the privacy of the polynomial against the verifier. But we can choose k large enough so that the privacy of Alice is virtually *perfect*. We will make this argument precise in the analysis of privacy of Info-Commit.

3.3 Secure Two-party computation based on $\binom{\mathbf{c}}{\mathbf{1}}_{\boldsymbol{q}}-$OT

A secure two-party computation algorithm $\mathbf{S2PC}(x, y, f, \mathcal{X}, \mathcal{Y}, \mathcal{F})$ enables two parties Alice and Bob to evaluate $f(x, y)$, where $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{F}$ is a publicly known function, $x \in \mathcal{X}$ is the private input of Bob and $y \in \mathcal{Y}$ is the private input of Alice [33]. At the end, Bob must learn $f(x, y)$ but nothing else about y, whereas Alice must learn nothing (not even $f(x, y)$). **S2PC** can be realized via an instantiation of $\binom{|\mathcal{X}|}{1}_{|\mathcal{F}|}-$OT as follows. The two parties first agree on an ordering of the elements of the set $\mathcal{X}$. Let ζ_i represent the i'th element of $\mathcal{X}$ in this ordering. Let $j \in [|\mathcal{X}|]$ be such that the private input of Bob satisfies $x = \zeta_j$. Bob can request the j'th row of Table 2 via an $\binom{|\mathcal{X}|}{1}_{|\mathcal{F}|}-$OT algorithm. Bob will learn $f(\zeta_j, y) = f(x, y)$ but nothing else about y, whereas Alice will learn nothing about x.

4 An Overview of Info-Commit

We start by noting that the polynomial $f(x) = a_0 + a_1 x + \cdots + a_{d-1} x^{d-1}$ can be rewritten as

$$f(x) = \left[1 \; x^s \; \cdots \; x^{s(s-1)}\right] A \left[1 \; x \; \cdots \; x^{s-1}\right]^T . \tag{7}$$

index	requested value
1	$f(\zeta_1, y)$
$\cdots$	$\cdots$
$\lvert\mathcal{X}\rvert$	$f(\zeta_{\lvert\mathcal{X}\rvert}, y)$

Table 2. In order to learn $f(x, y)$, the verifier will request the j'th row of this table via an $\binom{|\mathcal{X}|}{1}_{|\mathcal{F}|}-$OT algorithm where $j \in [|\mathcal{X}|]$ is such that $x = \zeta_j$.

Algorithm 1 Secure two-party computation, $\mathbf{S2PC}(x, y, f, \mathcal{X}, \mathcal{Y}, \mathcal{F})$

Input: Function $f : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathcal{F}$, Bob's private input $x \in \mathcal{X}$, Alice's private input $y \in \mathcal{Y}$.

Output: Bob learns $f(x, y)$ and nothing else about y. Alice will not learn anything.

1: The two parties agree on an ordering of the elements of the set $\mathcal{X}$. Let ζ_i represent the i'th element of $\mathcal{X}$ in this ordering.
2: Let $j \in [|\mathcal{X}|]$ be such that $x = \zeta_j$. The verifier requests the j'th row of Table 2 via an $\binom{|\mathcal{X}|}{1}_{|\mathcal{F}|}-$OT algorithm.

where $s = \sqrt{d}$ and

$$A = \begin{bmatrix} a_0 & a_1 & \cdots & a_{s-1} \\ a_s & a_{s+1} & \cdots & a_{2s-1} \\ & \ddots & & \\ a_{s^2-s} & a_{s^2-s+1} & \cdots & a_{s^2-1} \end{bmatrix}. \tag{8}$$

4.1 A basic algorithm

A rather straightforward algorithm is for the verifier to generate a $c \times s$ matrix $\Lambda \in \mathbb{F}_q^{c \times s}$ uniformly at random. In the commitment phase, the verifier and the prover will engage in a secure two-party computation algorithm to help the verifier learn

$$\Gamma = \Lambda A. \tag{9}$$

In this process, the verifier will not learn anything about A other than what is revealed through Γ, and the prover will not learn anything about Λ. In the evaluation phase, the verifier will ask the prover to compute $b = A \left[1 \; x \; \cdots \; x^{s-1}\right]^T$. Suppose the prover responds with $\hat{b}$. The verifier checks if $\Lambda\hat{b} = \Gamma \left[1 \; x \; \cdots \; x^{s-1}\right]^T$, which can be done in $O(\sqrt{d})$. If $\hat{b} \neq b$, this identity holds only with probability q^{-c} [28], since the prover does not know anything about Λ. If the verification process is successful, the verifier can recover $f(x) = \left[1 \; x^s \; \cdots \; x^{s(s-1)}\right] \hat{b}$ in $O(\sqrt{d})$. The main issue with this algorithm is that Γ reveals $\sqrt{d}$ symbols over $\mathbb{F}_q$ about the polynomial f during the commitment phase, and another $\sqrt{d}$ symbols for each evaluation b. We wish to reduce this to a constant. We will accomplish this in two steps: firstly, against a semi-honest verifier and next against a malicious verifier.

4.2 Improved privacy against a semi-honest verifier

In order to improve the privacy of this algorithm, the prover will generate a polynomial g of degree $d-1$ uniformly among all polynomials of degree $d-1$ over $\mathbb{F}_q$. In the commitment phase, instead of directly committing to f, the prover will commit to both $h = f + g$ and g. Let A and B represent the $s \times s$ matrices corresponding to the polynomials f and g, constructed similarly to (8). If both commitments have the same structure as in (9), the verifier will be able to gain substantial information about the matrix A by choosing his two secret matrices to be linearly dependent. To prevent this, the protocol will require the prover to commit to $A + B$ "from left" as in (9), whereas the commitment to B will be done from right. More concretely, the verifier and the prover will engage in a secure two-party computation algorithm to help the verifier learn

$$\begin{aligned} \Gamma &= \Lambda(A + B), \\ \Omega &= B\Theta^T, \end{aligned} \tag{10}$$

for two $c \times s$ secret matrices Λ and Θ generated randomly by the verifier. To gain intuition on how this helps with preserving the privacy of A, it helps to think of $c = 1$. Knowing any single linear combination of the rows of $A + B$ and any single linear combination of the columns of B cannot reveal more than one symbol over $\mathbb{F}_q$ about the elements of the matrix A. This argument will be made precise in the analysis of privacy of the Info-Commit. In the evaluation phase, the prover will be required to compute

$$\begin{aligned} v &= (A + B) \left[1 \; x \; \cdots \; x^{s-1}\right]^T, \\ u &= \left[1 \; x^s \; \cdots \; x^{s(s-1)}\right] B. \end{aligned} \tag{11}$$

Suppose the prover responds with $\hat{v}$ and $\hat{u}$. The verifier will check the correctness of each result via two parities

$$\begin{aligned} \Gamma \left[1 \; x \; \cdots \; x^{s-1}\right]^T &= \Lambda\hat{v}, \\ \left[1 \; x^s \; \cdots \; x^{s(s-1)}\right] \Theta^T &= \hat{u}\Omega. \end{aligned} \tag{12}$$

If both verification tests are successful, the verifier will recover

$$\begin{aligned} \hat{h}(x) &= \left[1 \; x^s \; \cdots \; x^{s(s-1)}\right] \hat{v}, \\ \hat{g}(x) &= \hat{u} \left[1 \; x \; \cdots \; x^{s-1}\right]^T. \end{aligned} \tag{13}$$

He will then find $\hat{f}(x) = \hat{h}(x) - \hat{g}(x)$ and will accept $\hat{f}(x)$ as the correct evaluation of $f(x)$. This algorithm guarantees the verifiability of the results with high probability following a similar analysis to [28]. Nonetheless, the privacy only holds as long as (Λ, Θ) are generated randomly, i.e., if the verifier is semi-honest. We will overcome this limitation in the next subsection.

4.3 Improving privacy against a malicious verifier

If both parties follow this protocol as described, the verifier will only learn a constant number of symbols about the polynomial f upon observing (Γ, Ω) and the pair (v, u) for a constant number of inputs (see the analysis of privacy in Section 6). However, a malicious verifier may deviate from this protocol by choosing Λ and Θ in a deterministic manner. To elaborate on this, suppose $c = 1$ and the verifier chooses Λ as the standard unit vector $\Lambda = \begin{bmatrix}1\ 0 \cdots 0\end{bmatrix}$. This helps him learn $\Gamma = A_1 + B_1$, the first row of the matrix $A + B$. Subsequently, in the evaluation phase, he requests the (v, u) pair for $x = 0$. Based on this, he can learn $u = B_1$, which is the first row of the matrix B. He will then proceed to subtract $\Gamma - u$ in order to find the first row of the matrix A, which reveals $\sqrt{d}$ symbols about the polynomial f.

We will propose a mechanism to resolve this threat. Suppose the verifier requests the (v, u) pair for m different inputs $x_1, \cdots, x_m$. Define the matrices X and Y as

$$X = \begin{bmatrix} 1 & x_1 & \cdots & x_1^{s-1} \\ & \cdots & \cdots & \\ 1 & x_m & \cdots & x_m^{s-1} \end{bmatrix},$$
$$Y = \begin{bmatrix} 1 & x_1^s & \cdots & x_1^{s(s-1)} \\ & \cdots & \cdots & \\ 1 & x_m^s & \cdots & x_m^{s(s-1)} \end{bmatrix}, \tag{14}$$

and let the matrices U and V be the concatenation of the vectors u and v for all $x \in \{x_1, \cdots, x_m\}$. In other words,

$$V = (A + B)X^T,$$
$$U = YB. \tag{15}$$

Since after m rounds of evaluation, the verifier learns both $\Gamma = \Lambda(A + B)$ and $U = YB$, the prover must ensure that the matrices Λ and Y do not contain any linear dependencies among their rows. Otherwise, this linear dependency can be exploited to extract substantial amount of information about the matrix A as outlined in the example above. Since $rank(Y) = m$ and $rank(\Lambda) = c$ (with high probability for randomly chosen Λ), the prover must be ensured that $rank(Y||\Lambda) = c + m$ where $||$ denotes the vertical concatenation of the two matrices. Similarly, we must have that $rank(X||\Theta) = c + m$. To accomplish this, we restrict the matrices Λ and Θ to have the same Vandermonde structures as the matrices Y and X, respectively. In other words, Λ and Θ must be of the

form

$$\Lambda = \begin{bmatrix} 1 & \lambda_1^s & \cdots & \lambda_1^{s(s-1)} \\ & \cdots & \cdots & \\ 1 & \lambda_c^s & \cdots & \lambda_c^{s(s-1)} \end{bmatrix},$$
$$\Theta = \begin{bmatrix} 1 & \theta_1 & \cdots & \theta_1^{s-1} \\ & \cdots & \cdots & \\ 1 & \theta_c & \cdots & \theta_c^{s-1} \end{bmatrix}. \tag{16}$$

Note that as long as

$$|\{\lambda_1, \cdots, \lambda_c, x_1, \cdots, x_m\}| = |\{\theta_1, \cdots, \theta_c, x_1, \cdots, x_m\}| = c + m, \tag{17}$$

the two rank requirements will be satisfied. Note also that without loss of generality, we can assume that $\lambda_i \neq \lambda_j$, $\theta_i \neq \theta_j$ and $x_i \neq x_j$ for $i \neq j$, since the verifier cannot benefit form receiving the same evaluation twice. Restricting the structure of Λ and Θ as opposed to choosing them uniformly at random over $\mathbb{F}_q^{c \times s}$ would mean that the prover now has some side information about these matrices which he can utilize to bypass the verification test. Fortunately, the analysis in Section 6 shows that this probability remains sufficiently small.

It is left to convince the prover that (17) holds. For this purpose, we designate a set $S \subseteq \mathbb{F}_q$ of prohibited values from which the elements λ_i and θ_i can be chosen. "Prohibited" means that in the evaluation phase, the verifier is not permitted to delegate the evaluation of the function f at any member of S to the prover. For instance, the verifier could provide an upper-bound ξ on its input values to the prover, and then $S \subseteq \mathbb{F}_q$ could be chosen as a sufficiently large interval whose smallest member is strictly larger than ξ.

4.4 Considerations regarding the size of the field

The privacy of Info-Commit as analyzed in Section 6 relies on the fact that the set $\{\alpha^s | \alpha \in S\}$ is of the same size as S where S is the set of prohibited values. This property holds if the function $g(x) = x^s$ is a permutation over $\mathbb{F}_q$ which is the case if $gcd(s, q-1) = 1$. It is also important that there exists a prohibited subset of $\mathbb{F}_q$ of sufficiently large size. To ensure this, either the verifier should set his upper bound ξ appropriately, or the field size should be increased once the verifier provides the upper-bound. Of course, this adjustment must be done without violating the first property.

5 Formal Description of Info-Commit

The formal description of the four functions of Info-Commit, namely **Commit**, **Eval**, **Verify** and **Recovery** are provided in Algorithms 2,3,4 and 5. Theorem 1 establishes that Info-Commit satisfies the requirements of an information-theoretically verifiable and private protocol for commitment and verification.

Theorem 1. *Info-Commit as described by the* **Commit**, **Eval**, **Verify** *and* **Recovery** *Algorithms 2,3,4 and 5 satisfies the correctness, soundness, privacy and efficiency requirements as stated in Definition 2. Specifically,*

- *If the prover is honest, the verifier will accept the results with probability 1.*
- *The probability that the prover can pass the verification process with a false result is negligible,*

$$\mathbb{P}(\mathbf{Verify}(x, \hat{val}, VK, K_v) = 1, \hat{val} \neq val) \leq \frac{2}{r^c}. \tag{18}$$

- *The verifier only learns $O(m^2)$ symbols about the coefficients of the polynomial f after he receives $(VK, val_1, \cdots, val_m)$ for any choice of K_v and any m input values $x_1, \cdots, x_m$.*

$$H_q(a_0, \cdots, a_{d-1} | \, VK, (x_1, val_1), \cdots, (x_m, val_m)) = d - O(m^2). \tag{19}$$

- *The complexity of the verifier per evaluation round is $O(\sqrt{d})$,*

$$\mathcal{C}_{\mathbf{Verify}} + \mathcal{C}_{\mathbf{Recovery}} = O(\sqrt{d}). \tag{20}$$

- *The complexity of the prover per evaluation round is $O(d)$,*

$$\mathcal{C}_{\mathbf{Eval}} = O(d). \tag{21}$$

Algorithm 2 The commitment function, $\mathbf{Commit}(a_0, \cdots, a_{d-1}, K_v, K_p)$

Input: The prover's private polynomial coefficients, $a_0, \cdots, a_{d-1}$, the verifier's secret key, K_v, the prover's secret key, K_p.
Output: The verifier learns the verification key VK$= (\Gamma, \Omega)$.

1: The verifier sends an upper-bound ξ on its maximum input to the prover. The two parties agree on a prohibited set $S \subset \mathbb{F}_q$ of size $r(s-1)$ where $s = \sqrt{d}$ and r is a small positive integer (for instance, $r = 10$). Every element of S must be greater than ξ.
2: The prover generates a random polynomial g of degree $d-1$ with matrix representation B. Let $h = f + g$ and $K_p = B$.
3: The verifier chooses c distinct elements of S uniformly at random, named $\lambda_1, \cdots, \lambda_c$ where c is a small positive integer (for instance, $c = 10$). Similarly, he chooses $\theta_1, \cdots, \theta_c$. Let $K_v = (\lambda_1, \cdots, \lambda_c, \theta_1, \cdots, \theta_c)$ and define Λ and Θ as in (16).
4: The two parties run c instances of $\mathbf{S2PC}(\alpha, A + B, \eta_1, S, \mathbb{F}^{s \times s}, \mathbb{F}^s)$ for $\alpha \in \{\lambda_i | i \in [c]\}$ where $\eta_1(\alpha, A + B) := \left[1 \; \alpha^s \; \cdots \; \alpha^{s(s-1)}\right](A + B)$. The verifier thus learns $\Gamma = \Lambda(A + B)$.
5: The two parties run c instances of $\mathbf{S2PC}(\alpha, B, \eta_2, S, \mathbb{F}^{s \times s}, \mathbb{F}^s)$ for $\alpha \in \{\theta_i | i \in [c]\}$ where $\eta_2(\alpha, B) := B\left[1 \; \alpha \; \cdots \; \alpha^{s-1}\right]^T$. The verifier thus learns $\Omega = B\Theta^T$.
6: Let VK$= (\Gamma, \Omega)$ be the verification key.

Algorithm 3 The evaluation function, **Eval**$(x, a_0, \cdots, a_{d-1}, K_p)$

Input: The coefficients of the polynomial, $a_0, \cdots, a_{d-1}$, the prover's secret key, $K_p = B$, the input to the polynomial, x.
Output: $val = (v, u)$.

The prover evaluates

$$\begin{aligned} v &= (A + B) \left[1 \; x \; \cdots \; x^{s-1}\right]^T, \\ u &= \left[1 \; x^s \; \cdots \; x^{s(s-1)}\right] B, \end{aligned} \tag{22}$$

and returns $val = (v, u)$ to the verifier.

Algorithm 4 The verification function, **Verify**$(x, \hat{val}, \mathrm{VK}, K_v)$

Input: The input to the polynomial, x, the evaluation provided by the prover, $\hat{val} = (\hat{v}, \hat{u})$, the (secret) verification key, VK$= (\Gamma, \Omega)$, and the verifier's secret key, K_v.
Output: The verifier either accepts or rejects $\hat{val}$.

The verifier checks the two parities

$$\begin{aligned} \Gamma \left[1 \; x \; \cdots \; x^{s-1}\right]^T &= \Lambda \hat{v}, \\ \left[1 \; x^s \; \cdots \; x^{s(s-1)}\right] \Theta^T &= \hat{u} \Omega. \end{aligned} \tag{23}$$

If either parity fails, then **Verify** returns 0 (rejected), otherwise, it returns 1.

Algorithm 5 The recovery function, **Recovery**$(x, \hat{val})$

Input: The input to the polynomial, x, and the evaluation provided by the prover, $\hat{val} = (\hat{v}, \hat{u})$.
Output: The evaluation of the polynomial $\hat{f}(x)$ based on $\hat{val}$.

If **Verify** returns 1, the verifier will compute $\hat{h}(x)$ and $\hat{g}(x)$ based on

$$\begin{aligned} \hat{h}(x) &= \left[1 \; x^s \; \cdots \; x^{s(s-1)}\right] \hat{v} \\ \hat{g}(x) &= \hat{u} \left[1 \; x \; \cdots \; x^{s-1}\right]^T. \end{aligned} \tag{24}$$

and finds $\hat{f}(x) = \hat{h}(x) - \hat{g}(x)$.

6 The Analysis of **Info-Commit**

Correctness. If the prover is honest, he will provide the evaluations of the same polynomial f that he has committed to. Consequently, in the verification phase, identities (23) trivially hold and the recovered value is equal to $h(x)-g(x) = f(x)$ as desired.
Efficiency.The verification can be done in $O(\sqrt{d})$ since it only requires multiplying $c \times s$ matrices by vectors of length $s = \sqrt{d}$. The recovery can be done in $O(\sqrt{d})$ since the main operation in this phase is inner products of vectors of length s. Therefore, the overall complexity of the verifier in the evaluation phase is $O(\sqrt{d})$. The prover, on the other hand, must compute the two vectors v and u according to (22) which can be done in $O(d)$.
Soundness. The analysis of soundness relies on the fact that the prover learns nothing about the secret values λ_i, θ_i, $i \in [c]$, during the commitment phase. This follows from the property that the oblivious transfer algorithms in [13, 14] provide perfect privacy for the receiver. Without loss of generality we can assume that the prover has committed to the correct polynomial f. If not, we simply use the letter f to denote the polynomial that the prover has committed to, and expect the soundness property to hold with respect to this f. Note that such a polynomial of degree $d-1$ exists regardless of how the prover responds to the queries in the commitment phase.

Remember that for each input x, the prover is required to provide the verifier with two vectors u and v as defined in (22). Here, we analyze the probability that a malicious prover can pass the verification test (23) with a $\hat{v} \neq v$ ($\hat{u} \neq u$) and show that this probability is negligible. Let $(\mathcal{V},\mathcal{U})$ denote a random variable from which the prover draws the evaluation $(\hat{v}, \hat{u})$. Note that $(\mathcal{V},\mathcal{U})$ is independent of (Λ, Θ). The probability that the prover can pass the first verification test with an incorrect result is given by

$$\begin{aligned} p_v &= \mathbb{P}\Big(\Gamma \left[1\ x \cdots\ x^{s-1}\right]^T) = \Lambda\mathcal{V}, \mathcal{V} \neq v\Big) \\ &= \mathbb{P}\Big(\Lambda(\mathcal{V} - v) = \mathbf{0}, \mathcal{V} \neq v\Big), \end{aligned} \tag{25}$$

where the randomness is with respect to $(\Lambda, \Theta, \mathcal{V},\mathcal{U})$. We need an upper-bound on p_v that holds for any x and any A, B. Define $\left[\gamma_0, \cdots, \gamma_{s-1}\right] = \mathcal{V} - v$. We can see this vector as the coefficients of a polynomial of degree $s-1$. The last equation is then the probability that the prover can provide a nontrivial polynomial of degree $s-1$ such that all λ_i^s, $i \in [c]$, are the roots of this polynomial. Note that the polynomial $\gamma(x) = \gamma_0 + \gamma_1 x + \cdots + \gamma_{s-1}x^{s-1}$ has at most $s-1$ roots in the set $S' = \{y^s | y \in S\}$. Note also that all the terms $\lambda_i^s, i \in [c]$, are distinct. This is because the field size q is chosen in such a way that $gcd(s, q-1) = 1$. As a result, the function $e(x) = x^s$ is a permutation, and the set S' has $r(s-1)$ distinct elements. Let $T \subseteq S'$ of size at most $|T| \leq s-1$ represent the roots of the polynomial $\gamma(x)$ that are in S'. In other words, $t \in T$, if and only if $\gamma(t) = 0$ and $t \in S'$. We are interested in the probability that $\lambda_i^s \in T$, $\forall i \in [c]$. Since $\mathcal{V}$ is

independent of Λ, so is the set T. we can therefore bound p_v as

$$p_v = \mathbb{P}(\lambda_i^s \in T, \forall i \in [c]) \leq \frac{\binom{s-1}{c}}{\binom{r(s-1)}{c}} \leq (\frac{s-1}{r(s-1)})^c = \frac{1}{r^c}. \tag{26}$$

Similarly, one can bound the probability that the prover can pass the second verification test with $\hat{u} \neq u$ as $p_u \leq \frac{1}{r^c}$. As a result,

$$\mathbb{P}(\textbf{Verify}(x, \hat{val}, \text{VK}, K_v) = 1, \hat{val} \neq val) \leq p_u + p_v \leq \frac{2}{r^c}. \tag{27}$$

For instance, by choosing $r = 10$ and $c = 10$, this probability can be reduced to 2×10^{-10}.
The next two lemmas will help us establish the privacy of Info-Commit.

Lemma 1. *Let c, d, s be three positive integers such that $c, d \leq s$. Let E be an $s \times s$ random matrix uniformly distributed over $\mathbb{F}_q^{s \times s}$. Let F and G be arbitrary full-rank $c \times s$ and $s \times d$ matrices respectively. Then*

$$H_q(FE|EG) \geq cs - cd. \tag{28}$$

Proof. Since, G is full-rank, we have that EG is uniformly distributed over $\mathbb{F}_q^{s \times d}$ and as a result, $H_q(EG) = sd$. So, we only need to prove that $H_q(FE, EG) \geq cs + sd - cd$. Let $L \in \mathbb{F}_q^{s^2}$ be a column vector obtained from the vertical concatenation of the columns of E, such that $L_{i+sj} = E_{i,j}$ for all $i \in [0 : s-1]$, $j \in [0 : s-1]$. Let $\mathcal{F} = \mathbf{I}_{s \times s} \otimes F$ where $\mathbf{I}_{s \times s}$ is the $s \times s$ identity matrix and $\otimes$ denotes the Kronecker product. Let $\mathcal{G} = G^T \otimes \mathbf{I}_{s \times s}$. Since the two vectors $\mathcal{F}L$ and $\mathcal{G}L$ are rearrangements of the the two matrices FE and EG, respectively, we have that $H_q(FE, EG) = H_q(\mathcal{F}L, \mathcal{G}L)$. The proof follows from the fact that the $(c+d)s$ by s^2 matrix $\mathcal{H} = \mathcal{F}||\mathcal{G}$ obtained from vertical concatenation of $\mathcal{F}$ and $\mathcal{G}$ has rank at least $(c+d)s - cd$ as long as F and G are full-rank (See Lemma 3 in the appendix).

Lemma 2. *Let E, F, G be three random variables with alphabets $\mathcal{E}, \mathcal{F}, \mathcal{G}$, satisfying the Markov chain $E \longleftrightarrow F \longleftrightarrow G$. Let $h : \mathcal{E} \times \mathcal{G} \to \mathcal{W}$ be an arbitrary function. Then, we have*

$$H(E|F, h(E, G)) \geq \min_{g \in \mathcal{G}} H(E|F, h(E, g)) \tag{29}$$

Proof. The following chain of inequalities prove the claim.

$$\begin{aligned} H(E|F, h(E, G)) &\geq H(E|F, G, h(E, G)) && (30)\\ &= \sum_{g \in \mathcal{G}} p(G = g) H(E|F, G = g, h(E, g)) && (31)\\ &\geq \min_{g \in \mathcal{G}} H(E|F, G = g, h(E, g)) = \min_{g \in \mathcal{G}} H(E|F, h(E, g)). && (32) \end{aligned}$$

Note that the first inequality is due to Property 2 of entropy terms in the preliminaries, and the last inequality follows from Property 4. Specifically, since $E \longleftrightarrow F \longleftrightarrow G$ forms a Markov chain, then $E \longleftrightarrow (F, h(E,g)) \longleftrightarrow G$ forms a Markov chain too, which allows us to drop $G = g$ from conditioning.

Privacy. We prove that after m rounds of evaluation, the verifier learns at most $(m+c)^2$ symbols over $\mathbb{F}_q$ about the matrix A which represents the polynomial f. The analysis of privacy relies on the fact that the verifier learns nothing about $(A+B, B)$ in the commitment phase except for what is implied via $(\mathit{\Gamma}, \mathit{\Omega})$. This follows because the secure two-party computation algorithm in [13, 14] provides perfect privacy for the sender except with negligible probability. The analysis also makes use of the fact that the λ_i and θ_i values in the commitment phase are chosen from a "prohibited" set. As a result of this, the matrices $\mathit{\Lambda}||Y$ and $\mathit{\Theta}||X$ will be full-rank. Note that the verifier cannot benefit from selecting $x_i = x_j$, i.e., requesting the same evaluation point twice. Similarly, he cannot benefit from setting $\lambda_i = \lambda_j$ or $\theta_i = \theta_j$. Therefore, without loss of generality, we can assume that $rank(\mathit{\Lambda}||Y) = rank(\mathit{\Theta}||X) = m + c$.

First, we assume that the privacy of the transmitter in the OT algorithm is perfect. At the end of the proof, we will show that the negligible probability that the privacy of the transmitter in the OT algorithm is compromised does not make a fundamental difference in our analysis. For (X, Y, V, U) defined as in (14),(15), we want to show that

$$H_q(A|V, U, \mathit{\Gamma}, \mathit{\Omega}) \geq d - (m+c)^2, \tag{33}$$

for any choice of the evaluation points $x_1, \cdots, x_m$ and for any choice of $\theta_1, \cdots, \theta_c$, $\lambda_1, \cdots, \lambda_c$. We start by simplifying the left hand side of (33).

$$\begin{aligned}
&H_q(A|V, U, \mathit{\Gamma}, \mathit{\Omega}) = H_q(A|(A+B)X^T, YB, \mathit{\Lambda}(A+B), B\mathit{\Theta}^T) \\
&= H_q(A, (A+B)X^T, \mathit{\Lambda}(A+B)|(A+B)X^T, YB, \mathit{\Lambda}(A+B), B\mathit{\Theta}^T) \\
&= H_q(A, BX^T, \mathit{\Lambda}B|(A+B)X^T, YB, \mathit{\Lambda}(A+B), B\mathit{\Theta}^T) \\
&= H_q(BX^T, \mathit{\Lambda}B|(A+B)X^T, YB, \mathit{\Lambda}(A+B), B\mathit{\Theta}^T) \\
&+ H_q(A|BX^T, \mathit{\Lambda}B, (A+B)X^T, YB, \mathit{\Lambda}(A+B), B\mathit{\Theta}^T) \\
&= H_q(BX^T, \mathit{\Lambda}B|(A+B)X^T, YB, \mathit{\Lambda}(A+B), B\mathit{\Theta}^T) + H_q(A|AX^T, \mathit{\Lambda}A),
\end{aligned} \tag{34}$$

where the last equality follows from the fact that A is independent of B. The second term in the final expression can be easily bounded as

$$\begin{aligned}
H_q(A|AX^T, \mathit{\Lambda}A) &= H_q(A) - H_q(AX^T, \mathit{\Lambda}A) \\
&\geq H_q(A) - H_q(AX^T) - H_q(\mathit{\Lambda}A) = d - ms - cs.
\end{aligned} \tag{35}$$

Therefore, we must show that the first term satisfies

$$H_q(BX^T, \mathit{\Lambda}B|(A+B)X^T, YB, \mathit{\Lambda}(A+B), B\mathit{\Theta}^T) \geq ms + cs - (m+c)^2. \tag{36}$$

Observe that $\Big((A+B)X^T, \Lambda(A+B)\Big)$ is independent of $\Big(BX^T, \Lambda B, YB, B\Theta^T\Big)$ (because $A+B$ is independent of B). So, by Property 3 of the entropy terms in the preliminaries, we have

$$H_q(BX^T, \Lambda B|(A+B)X^T, YB, \Lambda(A+B), B\Theta^T) = H_q(BX^T, \Lambda B|YB, B\Theta^T). \tag{37}$$

It is left to prove that $H_q(BX^T, \Lambda B|YB, B\Theta^T) \geq ms+cs-(m+c)^2$. Note that

$$\begin{aligned} &H_q(BX^T, \Lambda B|YB, B\Theta^T) = H_q(BX^T, \Lambda B, YB, B\Theta^T) - H_q(YB, B\Theta^T) \\ &= H_q(BX^T, B\Theta^T) + H_q(\Lambda B, YB|BX^T, B\Theta^T) - H_q(YB, B\Theta^T). \end{aligned} \tag{38}$$

Due to the fact that x_i and θ_i values are chosen from two different sets, we know that $\Theta||X$ is a full-rank matrix. Therefore, $H_q(BX^T, B\Theta^T) = (c+m)s$. Similarly, the matrix $\Lambda||Y$ is full-rank, thus by Lemma 1 we have $H_q(\Lambda B, YB|BX^T, B\Theta^T)$ $\geq (c+m)s-(c+m)^2$. Also, $H_q(YB, B\Theta^T) \leq H_q(YB) + H_q(B\Theta^T) = ms+cs$. Therefore, $H_q(BX^T, \Lambda B|YB, B\Theta^T) \geq ms+cs-(m+c)^2$. This gives us the desired inequality $H_q(A|V, U, \Gamma, \Omega) \geq d-(m+c)^2$.

We proved that $H_q(A|(A+B)X^T, YB, \Gamma, \Omega) \geq d-(m+c)^2$, for *every* choice of the evaluation points $x_1, \cdots, x_m$. Note that in general, the verifier may choose the evaluation points *after* observing the commitment (Γ, Ω). In other words, rather than assuming (X, Y) are arbitrary constants, we must treat them as random variables which satisfy the Markov chain $(A, B) \longleftrightarrow (\Gamma, \Omega) \longleftrightarrow (X, Y)$. But thanks to Lemma 2, since (Γ, Ω) appear in the conditioning, we have

$$H_q(A|(A+B)X^T, YB, \Gamma, \Omega) \geq \tag{39}$$

$$\min_{x_1, \cdots, x_m} H_q(A|(A+B)\begin{bmatrix} 1 & x_1 & \cdots & x_1^{s-1} \\ & \cdots & \cdots & \\ 1 & x_m & \cdots & x_m^{s-1} \end{bmatrix}^T, \begin{bmatrix} 1 & x_1^s & \cdots & x_1^{s(s-1)} \\ & \cdots & \cdots & \\ 1 & x_m^s & \cdots & x_m^{s(s-1)} \end{bmatrix} B, \Gamma, \Omega). \tag{40}$$

Therefore, the analysis above also addresses the case where the evaluation points are chosen adaptively, after observing the commitment.

Finally, let us show that the negligible probability that the privacy of the transmitter is compromised in the OT protocol does not make any fundamental difference in the analysis above. Let (Γ^*, Ω^*) represent the random variable that the verifier learns by the end of the commitment phase. The analysis in [31, 14, 13] indicates that with overwhelming probability, (Γ^*, Ω^*) does not reveal any more information about $(A+B, B)$ than what is implied through (Γ, Ω). Put differently, with overwhelming probability, the sequence $(\Gamma^*, \Omega^*) \longleftrightarrow (\Gamma, \Omega) \longleftrightarrow (A+B, B)$ forms a Markov chain. Let T be a binary random variable that is equal to 1 if and only if this Markov chain holds. We know that $P(T=0) = \epsilon$ (see for instance, Theorem 1 in [31]). Based on this, we can bound

$$\begin{aligned} &H_q(A|V, U, \Gamma^*, \Omega^*) \geq H_q(A|V, U, \Gamma^*, \Omega^*, T) \\ &= \mathbb{P}(T=1)H_q(A|V, U, \Gamma, \Omega) + \mathbb{P}(T=0)H_q(A|V, U, \Gamma^*, \Omega^*, T=0) \\ &\geq (1-\epsilon)H_q(A|V, U, \Gamma, \Omega). \end{aligned} \tag{41}$$

Note that ϵ can be made arbitrarily small by choosing a sufficiently large security parameter in the OT algorithm [31]. In particular, it can be made inversely proportional to d. Then

$$H_q(A|V, U, \Gamma^*, \Omega^*) \geq H_q(A|V, U, \Gamma, \Omega) - O(1). \tag{42}$$

Therefore, it suffices to show that $H_q(A|V, U, \Gamma, \Omega) = d - O(m^2)$ which we accomplished in the first part of the proof.

Acknowledgement

The authors would like to thank Mohammad Ali Maddah-Ali, Srivatsan Ravi and Ali Rahimi for the fruitful discussions and for revising the manuscript.

References

1. S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, "Delegating computation: interactive proofs for muggles," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*. ACM, 2008, pp. 113–122.
2. R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Annual Cryptology Conference*. Springer, 2010, pp. 465–482.
3. S. Benabbas, R. Gennaro, and Y. Vahlis, "Verifiable delegation of computation over large datasets," in *Annual Cryptology Conference*. Springer, 2011, pp. 111–131.
4. D. Fiore and R. Gennaro, "Publicly verifiable delegation of large polynomials and matrix computations, with applications," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 501–512.
5. B. Libert, S. C. Ramanna, and M. Yung, "Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions," in *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.
6. A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2010, pp. 177–194.
7. S. Li, M. Yu, S. Avestimehr, S. Kannan, and P. Viswanath, "Polyshard: Coded sharding achieves linearly scaling efficiency and security simultaneously," *arXiv preprint arXiv:1809.10361*, 2018.
8. B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable secret sharing and achieving simultaneity in the presence of faults," in *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*. IEEE, 1985, pp. 383–395.
9. Q. Zheng and S. Xu, "Secure and efficient proof of storage with deduplication," in *Proceedings of the second ACM conference on Data and Application Security and Privacy*. ACM, 2012, pp. 1–12.
10. J. Benet, D. Dalrymple, and N. Greco, "Proof of replication," *Protocol Labs*, 2017.
11. C. Cachin, C. Crépeau, and J. Marcil, "Oblivious transfer with a memory-bounded receiver," in *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*. IEEE, 1998, pp. 493–502.

12. Y. Aumann, Y. Z. Ding, and M. O. Rabin, "Everlasting security in the bounded storage model," *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1668–1680, 2002.
13. C. Cachin, C. Crépeau, J. Marcil, and G. Savvides, "Information-theoretic interactive hashing and oblivious transfer to a storage-bounded receiver," *IEEE Transactions on Information Theory*, vol. 61, no. 10, pp. 5623–5635, 2015.
14. Y. Z. Ding, D. Harnik, A. Rosen, and R. Shaltiel, "Constant-round oblivious transfer in the bounded storage model," in *Theory of Cryptography Conference*. Springer, 2004, pp. 446–472.
15. C. Papamanthou, E. Shi, and R. Tamassia, "Signatures of correct computation," in *Theory of Cryptography Conference*. Springer, 2013, pp. 222–242.
16. X. Ma, F. Zhang, and J. Li, "Verifiable evaluation of private polynomials," in *2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies*. IEEE, 2013, pp. 451–458.
17. X. Bultel, M. L. Das, H. Gajera, D. Gérault, M. Giraud, and P. Lafourcade, "Verifiable private polynomial evaluation," in *International Conference on Provable Security*. Springer, 2017, pp. 487–506.
18. D. Catalano and D. Fiore, "Vector commitments and their applications," in *International Workshop on Public Key Cryptography*. Springer, 2013, pp. 55–72.
19. J. Benaloh and M. De Mare, "One-way accumulators: A decentralized alternative to digital signatures," in *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 1993, pp. 274–285.
20. B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *2013 IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 238–252.
21. J. Groth, "On the size of pairing-based non-interactive arguments," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2016, pp. 305–326.
22. E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward, "Aurora: Transparent succinct arguments for r1cs," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2019, pp. 103–128.
23. H. Wu, W. Zheng, A. Chiesa, R. A. Popa, and I. Stoica, "{DIZK}: A distributed zero knowledge proof system," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 675–692.
24. E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 459–474.
25. M. Naor and B. Pinkas, "Oblivious polynomial evaluation," *SIAM Journal on Computing*, vol. 35, no. 5, pp. 1254–1281, 2006.
26. C. Hazay, "Oblivious polynomial evaluation and secure set-intersection from algebraic prfs," *Journal of Cryptology*, vol. 31, no. 2, pp. 537–586, 2018.
27. T. Tassa, A. Jarrous, and Y. Ben-Ya'akov, "Oblivious evaluation of multivariate polynomials," *Journal of Mathematical Cryptology*, vol. 7, no. 1, pp. 1–29, 2013.
28. S. Sahraei and A. S. Avestimehr, "INTERPOL: information theoretically verifiable polynomial evaluation," in *IEEE International Symposium on Information Theory (ISIT)*, 2019.
29. M. O. Rabin, "How to exchange secrets by oblivious transfer," *Technical Memo TR-81*, 1981.
30. S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Communications of the ACM*, vol. 28, no. 6, pp. 637–647, 1985.

31. Y. Z. Ding, "Oblivious transfer in the bounded storage model," in *Annual International Cryptology Conference*. Springer, 2001, pp. 155–170.

32. G. Brassard, C. Crepeau, and J.-M. Robert, "Information theoretic reductions among disclosure problems," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, 1986, pp. 168–173.

33. A. C. Yao, "Protocols for secure computations," in *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*. IEEE, 1982, pp. 160–164.

Appendix

Lemma 3. *Let c, d, s be three positive integers such that $c, d \leq s$. Let F and G be two full-rank $c \times s$ and $d \times s$ matrices over the field respectively. Let $\mathcal{F} = \mathbf{I}_{s\times s} \otimes F$ and $\mathcal{G} = G \otimes \mathbf{I}_{s\times s}$ and let $\mathcal{H} = \mathcal{F}||\mathcal{G}$ be the $(c+d)s$ by s^2 matrix obtained from vertical concatenation of $\mathcal{F}$ and $\mathcal{G}$. Then, $rank(\mathcal{H}) \geq (c+d)s - cd$.*

Proof. Since F is full-rank and $c \leq s$, F must have at least one $c \times c$ full-rank submatrix. Let $\tau \subseteq [0 : s-1]$ represent the indices of the columns of one such sub-matrix and let F_τ be the corresponding submatrix of F. Define

$$\rho = \{cs + i + sj | j \in [0 : d-1], i \in \tau\}. \tag{43}$$

Intuitively, ρ corresponds to the rows of $\mathcal{G}$ which may have a non-zero element in any column from τ. Note that $|\rho| = cd$. we will argue that by eliminating the rows indexed in ρ from $\mathcal{H}$ we will find a full-rank matrix. Since this resulting matrix has $(c+d)s - cd$ rows, the claim will follow. In Equation (44) below, we have illustrated an example with $s = 4, c = 3$ and $d = 2$. We have assumed that $\tau = \{1, 2, 3\}$ and have marked the full-rank submatrix of F in blue. The rows indexed in ρ are shown in red.

Let λ be a vector of length $(c+d)s$ such that $\lambda_i = 0, \forall i \in \rho$ and $\lambda\mathcal{H} = \mathbf{0}$. We will show that λ must be the all-zero vector. For $j \in [0 : s-1]$, let $\mathcal{H}_j$ be the submatrix of $\mathcal{H}$ obtained from columns $\{i+sj | i+cs \in \rho\}$. Since $\lambda\mathcal{H} = \mathbf{0}$, we must have $\lambda\mathcal{H}_j = \mathbf{0}$. But for any $i \in [0 : (c+d)s-1] \backslash [jc : (j+1)c-1]$, either $\lambda_i = 0$ or the i'th row of $\mathcal{H}_j$ is an all-zero vector. It follows that $\lambda_{[jc:(j+1)c-1]} F_\tau = \mathbf{0}$. But F_τ is full-rank, so $\lambda_{[jc:(j+1)c-1]} = \mathbf{0}$. Since this holds for every $j \in [0 : s-1]$, we conclude that $\lambda_{[0:cs-1]} = \mathbf{0}$.

It follows from $\lambda_{[0:cs-1]} = \mathbf{0}$ and $\lambda\mathcal{H} = \mathbf{0}$ that $\lambda_{[cs:(c+d)s-1]}\mathcal{G} = \mathbf{0}$. Define the $s \times d$ matrix θ such that $\theta_{i,j} = \lambda_{i+sj}$. Observe that $\lambda_{[cs:(c+d)s-1]}\mathcal{G}$ is a vector found by rearranging the elements of the matrix θG. Since $\lambda_{[cs:(c+d)s-1]}\mathcal{G} = \mathbf{0}$, we must have $\theta G = \mathbf{0}$. Since G is full-rank, it follows that $\theta = \mathbf{0}$ and as a result

$\lambda_{[cs:(c+d)s-1]} = \mathbf{0}$. We conclude that $\lambda = \mathbf{0}$ which establishes the claim.

$$\mathcal{H} = \begin{bmatrix}
f_{0,0} & f_{0,1} & f_{0,2} & f_{0,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
f_{1,0} & f_{1,1} & f_{1,2} & f_{1,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
f_{2,0} & f_{2,1} & f_{2,2} & f_{2,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & f_{0,0} & f_{0,1} & f_{0,2} & f_{0,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & f_{1,0} & f_{1,1} & f_{1,2} & f_{1,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & f_{2,0} & f_{2,1} & f_{2,2} & f_{2,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & f_{0,0} & f_{0,1} & f_{0,2} & f_{0,3} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & f_{1,0} & f_{1,1} & f_{1,2} & f_{1,3} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & f_{2,0} & f_{2,1} & f_{2,2} & f_{2,3} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & f_{0,0} & f_{0,1} & f_{0,2} & f_{0,3} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & f_{1,0} & f_{1,1} & f_{1,2} & f_{1,3} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & f_{2,0} & f_{2,1} & f_{2,2} & f_{2,3} \\
g_{0,0} & 0 & 0 & 0 & g_{0,1} & 0 & 0 & 0 & g_{0,2} & 0 & 0 & 0 & g_{0,3} & 0 & 0 & 0 \\
0 & g_{0,0} & 0 & 0 & 0 & g_{0,1} & 0 & 0 & 0 & g_{0,2} & 0 & 0 & 0 & g_{0,3} & 0 & 0 \\
0 & 0 & g_{0,0} & 0 & 0 & 0 & g_{0,1} & 0 & 0 & 0 & g_{0,2} & 0 & 0 & 0 & g_{0,3} & 0 \\
0 & 0 & 0 & g_{0,0} & 0 & 0 & 0 & g_{0,1} & 0 & 0 & 0 & g_{0,2} & 0 & 0 & 0 & g_{0,3} \\
g_{1,0} & 0 & 0 & 0 & g_{1,1} & 0 & 0 & 0 & g_{1,2} & 0 & 0 & 0 & g_{1,3} & 0 & 0 & 0 \\
0 & g_{1,0} & 0 & 0 & 0 & g_{1,1} & 0 & 0 & 0 & g_{1,2} & 0 & 0 & 0 & g_{1,3} & 0 & 0 \\
0 & 0 & g_{1,0} & 0 & 0 & 0 & g_{1,1} & 0 & 0 & 0 & g_{1,2} & 0 & 0 & 0 & g_{1,3} & 0 \\
0 & 0 & 0 & g_{1,0} & 0 & 0 & 0 & g_{1,1} & 0 & 0 & 0 & g_{1,2} & 0 & 0 & 0 & b_{1,3}
\end{bmatrix}. \tag{44}$$